THESIS

APPLICATION OF ARTIFICIAL BOUNDARY CONDITIONS IN SENSITIVITY-BASED UPDATING OF FINITE ELEMENT MODELS

by

John R. Mentzer

June 2007

Thesis Advisor: Joshua H. Gordis

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Application of Artificial Boundary Conditions in Sensitivity-Based Updating of Finite Element Models.			**5. FUNDING NUMBERS**
6. AUTHOR(S) John R. Mentzer			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

In structural dynamics the ability of a finite element model, or (FEM), to accurately represent a structure's dynamic response (natural frequencies and mode shapes) determines its utility as a solution tool. Often the model needs to be updated or improved to better represent the structure it is modeling. An updated or improved model of an undamaged structure is often needed in order to identify damage in an in-service structure. A difficulty generally arises in trying to solve for this error because it is often represented by an underdetermined problem, as the number of parameters potentially in error in the FEM is typically much larger than the number of measured parameters. The method of Artificial Boundary Conditions (ABC) can help to resolve the problem and lead to an improved solution. The ABC systems provide the natural frequencies for the structure under test, under a variety of boundary conditions which are imposed computationally. Specifically, the use of ABC in sensitivity based updating will be investigated and its improvement on performance reviewed.

14. SUBJECT TERMS Artificial Boundary Conditions Finite Element Mode Updating			**15. NUMBER OF PAGES** 111
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

APPLICATION OF ARTIFICIAL BOUNDARY CONDITIONS IN SENSITIVITY-BASED UPDATING OF FINITE ELEMENT MODELS.

John R. Mentzer
Lieutenant, United States Navy
B.S., Naval Architecture, United States Naval Academy, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2007

Author: John R. Mentzer

Approved by: Joshua H. Gordis
Thesis Advisor

Anthony J. Healey
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In structural dynamics the ability of a finite element model, or (FEM), to accurately represent a structure's dynamic response (natural frequencies and mode shapes) determines its utility as a solution tool. Often the model needs to be updated or improved to better represent the structure it is modeling. An updated or improved model of an undamaged structure is often needed in order to identify damage in an in-service structure. A difficulty generally arises in trying to solve for this error because it is often represented by an underdetermined problem, as the number of parameters potentially in error in the FEM is typically much larger than the number of measured parameters. The method of Artificial Boundary Conditions (ABC) can help to resolve the problem and lead to an improved solution. The ABC systems provide the natural frequencies for the structure under test, under a variety of boundary conditions which are imposed computationally. Specifically, the use of ABC in sensitivity based updating will be investigated and its improvement on performance reviewed.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would to thank Professor Gordis for his seemingly inexhaustible patience and academic encouragement. To Amanda, I cannot thank you enough for all of your support and encouragement throughout this process.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A finite element model, (FEM) is a computational representation of a structure or object that is generally used to solve a complex problem for which a closed form solution is unavailable. The nature of the problem frequently means that these structures or objects are very complex in design. For many systems, the FEM is the only means by which the response of the system can be calculated.

The FEM requires a number of parameters, including, but not limited to, physical geometry, density and modulus of elasticity. Although a structure is defined by a large number of parameters only a small number of parameters can be measured in a modal test. It is these measured parameters from a modal test are the modal parameters of a structure, i.e., natural frequency and mode shapes. It is these parameters that define a structure and will be the focus of this paper. The ability of a model to accurately predict the modal parameters is of vital importance. Often the model must be corrected or updated in order to accurately reflect the correct parameters. To update the modal real world data must be applied to and correct the FEM.

In most real world problems a limited number of measured modal parameters from the structure creates an underdetermined problem. This underdetermined problem is a product of the large number of adjustable parameters compared to the relatively small number of measured parameters. Often the number of measured parameters are limited due to equipment limitations (i.e., inability to measure modes at high frequencies). However, when measuring and obtaining data from a structure, the number of measurements can also be limited by the number of measuring devices available (i.e., accelerometers). Thus, it is necessary to place the devices in either locations of interest or locations of ease. These measured locations result in the “analysis set” of coordinates or “ASET”. The remaining set is referred to as the “omitted set” or “OSET”. Due to most finite element models having a large number of coordinates, or degrees of freedom (DOF), the number of OSET coordinates is much greater than the number of ASET coordinates.

One method of solving the above problem is to obtain more measured parameters from the structure. One approach to this is to obtain modal parameters for a larger number of modes. This is often prohibitive due to the difficulties associated with measuring higher frequency modes for a structure. Another approach involves the application of additional/new boundary conditions to the structure and obtaining more data. However, this is often costly or simply impossible. Thus, a more feasible solution method is desired. One option is the method of Artificial Boundary Conditions.

The method of Artificial Boundary Conditions applies a pseudo boundary condition to the structure and extracts modal parameters from the pre-existing data. The term pseudo is used because an actual physical boundary condition is not applied, the model merely reacts as if additional degrees-of-freedom (DOF) are restrained. This results in the ability to extract additional modal parameters without having to conduct additional testing on the structure. When applying this method to model updating it can often help to resolve an underdetermined problem.

Sensitivity based updating makes use of the modal parameters to help locate a difference between a FEM and an existing structure. This difference could exist for a variety of reasons. The difference could be due to damage or an unknown alteration of the structural system that is desired to be located. The difference could also be the result of a poorly representative model, or even the result of a model that is not capable replicating the physics of the structure. Trying to locate this difference is a problem that is generally underdetermined due to the number of parameters that can be altered and the limited number of measured parameters. With the application of ABC a better solution to this problem can often be formulated.

II. THEORY

The general equation of motion will be the fundamental starting point for the problem.

$$[M]\left\{\ddot{x}\right\}+[C]\left\{\dot{x}\right\}+[K]\{x\}=\{F(t)\} \tag{1}$$

This problem consists of mass, damping and stiffness parameters, where $[K]$, $[M]$ and $[C]$ are the symmetric stiffness, mass, and damping matrices, of size n x n, and $\{F(t)\}$ is the excitation vector of size nx1. They can be more explicitly stated in terms of their components within each matrix, and are so stated below.

$$\begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix}\begin{Bmatrix} \ddot{x}_1(t) \\ \\ \ddot{x}_n(t) \end{Bmatrix}+\begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}\begin{Bmatrix} \dot{x}_1(t) \\ \\ \dot{x}_n(t) \end{Bmatrix}+\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix}\begin{Bmatrix} x_1(t) \\ \\ x_n(t) \end{Bmatrix}=\begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2}$$

The above equation is for *n* number of DOF. As mention previously, in real world analysis, it is improbable that all DOF will be instrumented and recorded. In this testing the group of DOF that are instrumented and recorded are considered the analytical set or “ASET” while the remaining DOF are considered the omitted or “OSET”.

A. ANALYSIS AND OMITTED COORDINATE SETS

When conducting modal testing the system being analyzed can be broken up into two types of coordinate sets. Starting with the EOM listed above and applying a simple harmonic excitation

$$\{F(t)\}=\left\{\bar{F}\right\}e^{j\Omega t} \tag{3}$$

Giving a steady state response of

$$\{x(t)\}=\left\{\bar{X}\right\}e^{j\Omega t} \tag{4}$$

Now substituting this into the EOM for the *n* DOF system.

$$\left[\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix} + j\Omega \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}\right] + \begin{Bmatrix} x_1(t) \\ \\ x_n(t) \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \quad (5)$$

Taking this example and now thinking if it in terms of ASET and OSET coordinate systems we can rewrite the $[K]$, $[M]$, and $[C]$ in those terms, leading to

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix} + j\Omega \begin{bmatrix} C_{aa} & C_{ao} \\ C_{oa} & C_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ f_o \end{Bmatrix} \quad (6)$$

Looking at the above, there obviously exists a relationship between the ASET and OSET coordinates in which the ASET coordinates can be used to develop the solution for the OSET coordinates (Gordis, 1993). Ignoring damping and assuming that there is no excitation acting on the OSET coordinate system then equation can be written as

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ 0 \end{Bmatrix} \quad (7)$$

The matrices can easily be broken into the two separate equations. Looking at the two equations, the solution can be determined for the OSET coordinates in terms of ASET coordinates using just one of the two.

$$[K_{oa}]\{x_a\} + [K_{oo}]\{x_o\} - \Omega^2\left[[M_{oa}]\{x_a\} + [M_{oo}]\{x_o\}\right] = 0 \quad (8)$$

Then grouping the similar terms and solving in terms of {x$_o$}, Eqn. (8) becomes

$$\{x_o\} = \left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1}\left[-K_{oo}^{-1}K_{oa} + \Omega^2 K_{oo}^{-1} M_{oa}\right]\{x_a\} \quad (9)$$

In Eqn. (9) the $\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1}$ term can be written

$$\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1} = \frac{1}{Det\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]} Adj\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right] \quad (10)$$

where Det[…] indicates the determinate and Adj[…] indicates the adjoint matrix. From Eqn. (10) it is evident that the solution for $\{x_o\}$ in Eqn. (9) does not exist at those frequencies, which satisfy

$$Det\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right] = 0 \tag{11}$$

The frequencies which satisfy Eqn. (11) are the eigenvalues of the system defined by the matrices [K_{oo}] and [M_{oo}]. Since both of these matrices are composed solely of OSET coordinates, the resulting system is considered the OSET system and the resulting frequencies are the OSET frequencies. Knowing that the eigenvector solutions to a system are defined by the unrestrained DOF it should be noticed that the OSET system is considered the unrestrained DOF and the ASET the restrained or grounded set.

The use of a limited number of analyzed DOF to represent a complete structure is imperative in structural dynamics. Even if the number of ASET coordinates is relatively large it would still be far less than the literally infinite OSET coordinates (Gordis, 1999). Thus even though a modal test is limited in the number of ASET coordinates, it is imperative that the use of this incomplete coordinate/data set be used to represent a full system.

B. SPATIALLY INCOMPLETE DATA

As mentioned previously a finite element model can represent a real world system with an infinite number of DOF. However, real world limitations allow only a limited number of DOF to be measured in a modal test of a system. This lack of a complete data for all DOF creates a spatially incomplete problem. This leads to the necessity of finding a solution using the spatially incomplete data set. One method to do this is using the spatially incomplete data to solve for the OSET frequencies which will be derived in the next section. An understanding of how this spatially incomplete data compares to spatially complete data follows.

Consider first the complete FRF matrix, which is *n x n* and then suppose that the description of the system is limited to only certain measured coordinates, thus ignoring what happens at the other coordinates. (Note that ignoring coordinates is not the same as supposing that the other coordinates do not exist.) The resulting model is now of the order N x N. It is clear that because the basic system has not been altered and that it still has the same number of DOF, even though it was decided not to describe all of the DOF, the elements which remain in the reduced FRF matrix are identical to the corresponding elements in the full *n x n* matrix. (Ewins, 1982) Essentially this is saying that the reduced matrix is composed of elements of interest from the original matrix and leaving behind those that are ignored. This matrix however still contains all the modal information of a fully described matrix.

Specifically looking at the before mentioned description of a reduced matrix and that the data contained in a measured FRF matrix, which is composed of response information from a limited number of measurements, it can be seen that it contains all of the information of the entire system. Also given that the measured FRF matrix implicitly defines a dynamically reduced impedance model the reduction of the FE model is pursued is the same manner (Gordis, 1996). Therefore the use of a spatially incomplete data set to solve for the entire system is justified. This is of particular significance in the application of FRFs.

III. FREQUENCY RESPONSE FUNCTION

A Frequency Response Function, or FRF, is the ratio of the output response of a structure due to an applied force. The FRF contains the amplitude and phase for the response for a specific DOF due to an input force of unit amplitude at a specific DOF. This amplitude and phase describe where the natural frequencies of a system lie and are of great use in model updating. Additionally FRF data can be estimated from output only measurements on operating machinery (Hanson, 2005). Due to these attributes the use of FRF in structural dynamic modal testing is widespread.

An FRF is constructed by measuring both the applied force and the response of the structure to the applied force simultaneously. The measured data are then transformed from the time domain to the frequency domain using Fast Fourier Transform (FFT) algorithms. Due to this transformation the function ends up being constructed in terms of real and imaginary components or magnitude and phase components.

The FRF matrix, $\left[H(\Omega)\right]$, relates the amplitude of stead state harmonic forcing at DOF "*j*" to the amplitude of the steady state harmonic response at DOF "*i*". The inverse of the FRF matrix is known as the impedance matrix $\left[Z(\Omega)\right]$.

$$\left[Z(\Omega)\right] = \left[H(\Omega)\right]^{-1} \tag{12}$$

where

$$\left[Z(\Omega)\right] = \left[K - \Omega^2 M + j\Omega C\right] \tag{13}$$

A. REDUCED ORDER MODEL

The limited number of instruments recording data in a model describes a reduced order model. This reduced order model's impedance is non-linearly dependent on the impedance of the full order model (Gordis, 1999). A full impedance matrix of infinite DOF is described below

$$[H] = \begin{bmatrix} H_{aa} & H_{ao} \\ H_{oa} & H_{oo} \end{bmatrix} \qquad (14)$$

This is the combination of measured coordinates, also considered experimental set, a and omitted coordinate set, o. The experimental set can also be thought of as the reduced model indicated by the overbar as shown below.

$$\left[\overline{H^x}\right] = [H_{aa}] \qquad (15)$$

From the previous discussion of the impedance and FRF matrix it is known that

$$\begin{bmatrix} Z_{aa} & Z_{oa} \\ Z_{ao} & Z_{oo} \end{bmatrix} \begin{bmatrix} H_{aa} & H_{oa} \\ H_{ao} & H_{oo} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (16)$$

Writing the matrix in equation form

$$\begin{aligned} Z_{aa}H_{aa} + Z_{ao}H_{oa} &= 1 \\ Z_{oa}H_{aa} + Z_{oo}H_{oa} &= 0 \end{aligned} \qquad (17)$$

Now solving for $[H_{aa}]$ gives

$$[H_{aa}] = \left[Z_{aa} - Z_{ao}Z_{oo}^{-1}Z_{oa}\right]^{-1} \qquad (18)$$

Knowing that $[Z] = \left[K - \Omega^2 M - j\Omega C\right]$ or $\left[K - \Omega^2 M\right]$ if damping is zero and that the eigenvalue solution is defined by the solution to $\left[K - \Omega^2 M\right]$=0 it can be seen that the solution to $[H_{aa}] = \left[Z_{aa} - Z_{ao}Z_{oo}^{-1}Z_{oa}\right]^{-1}$ is defined by the Z_{oo}^{-1} term. This shows that elements of H_{aa}^{-1} will be singular at the OSET natural frequencies (Gordis, 1999). To show the importance of this singularity a brief derivation of the FRF follows.

B. DERIVATION OF FREQUENCY RESPONSE FUNCTION

Looking at a impedance matrix in terms of an *n* DOF system

$$\begin{bmatrix} Z_{11} & \cdots & Z_{1n} \\ \vdots & \ddots & \vdots \\ Z_{n1} & \cdots & Z_{nn} \end{bmatrix} \begin{Bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \qquad (19)$$

Then in terms of the FRF matrix for the same *n* DOF system

$$\begin{Bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{Bmatrix} = \begin{bmatrix} H_{11} & \cdots & H_{1n} \\ \vdots & \ddots & \vdots \\ H_{n1} & \cdots & H_{nn} \end{bmatrix} \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{20}$$

Knowing that $\{x\} = [\Phi]\{q\}$, where Φ is the mass normalized mode shapes of the system and *q* gives the generalized coordinate set

$$[Z][\Phi]\{q\} = \{f\} \tag{21}$$

Then pre-multiplying both sides by $[\Phi]^T$

$$[\Phi]^T [Z][\Phi]\{q\} = [\Phi]^T \{F\} \tag{22}$$

Then expanding $[Z]$ from Eqn (13).

$$\Big[[\Phi]^T [K][\Phi] - \Omega^2 [\Phi]^T [M][\Phi] + j\Omega[\Phi]^T [C][\Phi]\Big]\{q\} = [\Phi]^T \{F\} \tag{23}$$

Invoking that Eqn. (23) is mass normalized $[\Phi]^T [M][\Phi] = 1$ and assuming proportional damping, $\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix}$ Eqn (23) reduces to

$$\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]\{q\} = [\Phi]^T \{F\} \tag{24}$$

Where ω_i ω_i is the *i*th natural frequency of the system, ζ is the systems damping ratio, and Ω is the forcing frequency. At this point $\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]$ is known as the modal impedance matrix and is diagonal.

Inverting the modal impedance matrix to find the modal frequency response matrix gives

$$\{x\} = [\Phi] \frac{1}{\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]} [\Phi]^T \{F\} \tag{25}$$

Transforming Eqn. (25) back into physical coordinates, and pre-multiplying by $\left[\Phi\right]$ and making use of $\left\{\Im\right\} = \left[\Phi\right]^T \left\{F\right\}$

$$\left\{x\right\} = \left[\Phi\right] \frac{1}{\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]} \left[\Phi\right]^T \left\{F\right\} = \left[\Phi\right]\left\{q\right\} \tag{26}$$

From Eqn. (26) $\left[H(\Omega)\right]$ can be extracted and defined as

$$\left[H(\Omega)\right] = \left[\Phi\right] \frac{1}{\left[\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i\right]} \left[\Phi\right]^T \tag{27}$$

The FRF Matrix can also be written in summation form for each element.

$$\left[H(\Omega)\right] = \sum_{k=1}^{\text{modes}} \frac{\left\{\Phi^k\right\}\left\{\Phi^k\right\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k} \tag{28}$$

This summation form can be descriptive in terms of $\left[H_{ij}(\Omega)\right]$.

$$\left[H_{ij}(\Omega)\right] = \sum_{k=1}^{\text{modes}} \frac{\left\{\Phi_i^k\right\}\left\{\Phi_j^k\right\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k} \tag{29}$$

From Eqn. (29) it can be seen that the FRF contains the mode shape data for the system. In fact it should be noted that any row of the FRF contains all the mode shape data for the entire system.

The FRF output display in the frequency domain also contains and displays the resonance and anti-resonance frequencies of the system. It is this resonance and anti-resonance frequency information that will be of interest in later sections. It should be noted that the subscripts for the mode shapes indicate the DOF that the force is being applied (j), and response at DOF measured (i). When these two are the same DOF is termed the driving point, or "point mobility", function and has some unique characteristics. When (i) and (j) are different DOFs it is termed "transfer mobility" function (Ewins, 1982).

The following example is a two DOF system with no damping. This example comes from Ewins, 1982. and displays the characteristics of both a driving point and transfer function for the complete FRF.

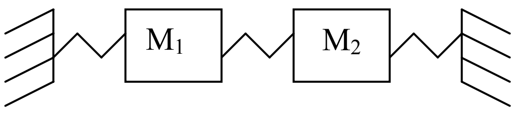


Figure 1. 2 DOF Example

With the stiffness matrix defines as $\left[K\right]=\begin{bmatrix} K_1+K_2 & -K_2 \\ -K_2 & K_2+K_3 \end{bmatrix}$

and the mass matrix $\left[M\right]=\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}$

For this example $M_1 = M_2 = 1.0$ and $K_1 = K_3 = 0.4$ and $K_2 = .8$. This yields the following:

Mode shapes: $\left\{\Phi^1\right\} = \begin{Bmatrix} -0.7071 \\ -0.7071 \end{Bmatrix}$ $\left\{\Phi^2\right\} = \begin{Bmatrix} -0.7071 \\ 0.7071 \end{Bmatrix}$

Natural frequencies: $\left\{\omega(rad/\sec)\right\} = \begin{Bmatrix} 0.6325 \\ 1.0954 \end{Bmatrix}$

Applying this with Eqn. (29) $\left[H_{ij}(\Omega)\right] = \sum_{k=1}^{\text{modes}} \frac{\left\{\Phi_i^k\right\}\left\{\Phi_j^k\right\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k}$

For the driving point function

$$\left[H_{11}(\Omega)\right] = \frac{\left\{\Phi_1^1\right\}\left\{\Phi_1^1\right\}^T}{\omega_1^2 - \Omega^2} + \frac{\left\{\Phi_1^2\right\}\left\{\Phi_1^2\right\}^T}{\omega_2^2 - \Omega^2} = \frac{\left\{-0.7071\right\}\left\{-0.7071\right\}^T}{0.4 - \Omega^2} + \frac{\left\{-0.7071\right\}\left\{-0.7071\right\}^T}{1.2 - \Omega^2} \qquad (30)$$

and for the transfer function.

$$\left[H_{12}(\Omega)\right] = \frac{\left\{\Phi_1^1\right\}\left\{\Phi_2^1\right\}^T}{\omega_1^2 - \Omega^2} + \frac{\left\{\Phi_1^2\right\}\left\{\Phi_2^2\right\}^T}{\omega_2^2 - \Omega^2} = \frac{\left\{-0.7071\right\}\left\{-0.7071\right\}^T}{0.4 - \Omega^2} + \frac{\left\{-0.7071\right\}\left\{0.7071\right\}^T}{1.2 - \Omega^2} \qquad (31)$$

For the driving point function it can be seen that the two terms will have the same sign for forcing frequencies above and below the two natural frequencies (resonance peaks). This creates an additive effect where the two terms are summed to create a larger third. This is displayed in the top portion of Figure (2) which is the Driving Point FRF $\left[H_{11}\right]$. However, in between the two natural frequencies the two terms are of opposite sign and thus are subtractive in nature. Due to this there exists a forcing frequency that will cause the terms to equal each other and thus create an antiresonance. It should be noted that the "y" axis is log based thus the addition and subtraction of the two terms does not fall equally between them on the plot

Looking at the transfer function it is evident that the two terms will have opposite signs above and below the two natural frequencies. This creates a subtractive effect in the two regions. This is shown in the lower plot of Figure (2) which is the H_{12}. Conversely in the region between the two natural frequencies the two terms are additive and thus do not create an antiresonance.

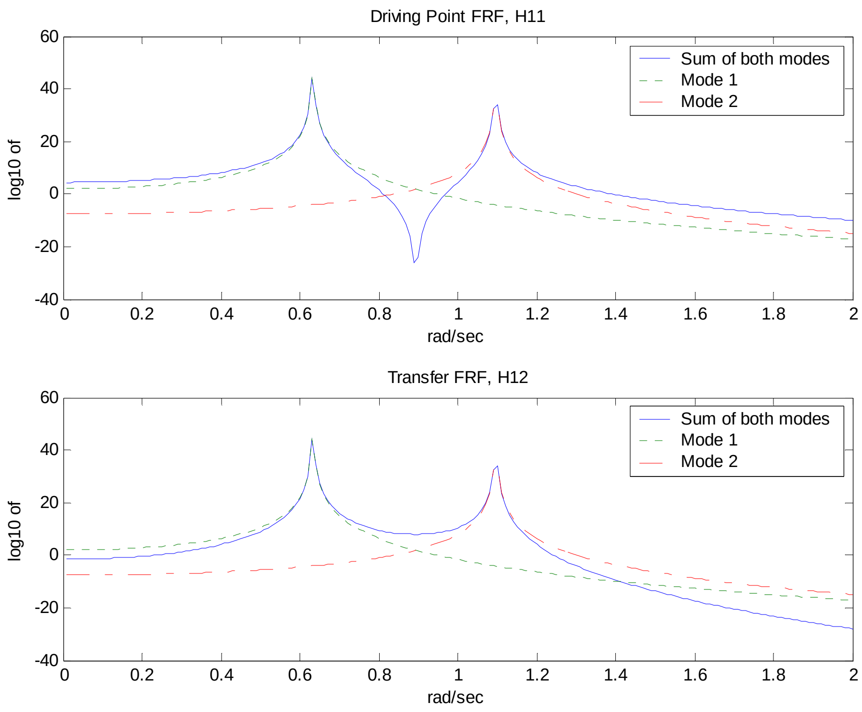


Figure 2. 2 DOF Frequency Response Function

This principles demonstrated here can be extended to any number DOF and thus create a fundamental rule that has great value. That is that if two consecutive modes have the same sign for the modal constants, then there will be an anti-resonance frequency between the two natural frequencies of those two modes (Ewins, 1982). This concept of the existence of an antiresonance between any two modes of a driving point FRF is of great significance in the use of Artificial Boundary Conditions.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ARTIFICIAL BOUNDARY CONDITIONS

As discussed previously the ability of a finite element model to accurately represent the dynamic behavior of a structure is the desired end result. Of particular importance are the similarities between the natural frequencies of the model and the actual structure. Often there exists a disparity between the two and it is desired to correct or "update" the model to better represent the structure. In order to do this it is necessary to obtain as much data from modal testing as possible. This could be resolved by conducting multiple tests of the system under different boundary conditions. This would obviously be a very expensive and time consuming procedure. The use of ABC can alleviate the need for additional testing by mathematically manipulating the existing data.

A. ARTIFICIAL BOUNDARY CONDITIONS DEFINED

The term artificial boundary condition describes the constraining or "pinning" of a specified DOF on a finite element model. The term artificial portrays that there is no actual physical boundary conditions applied to the structure under test. However, the test data from the structure is imposed with computational constraints. The existing model is merely updated with new boundary conditions that are easily imposed on a finite element model. This then generates additional data that can be used in the comparison of the two. Numerous ABC can be applied to a single FE model thus generating several models varying only in the boundary conditions. The new boundary condition provides a new configuration to the model, yet only one set of measured test data is required. These ABC are the boundary conditions that define the OSET. As previously shown OSET provide additional frequency information about the model and this additional information can help to eliminate ill-conditioning in a solution (Gordis, 1999).

A simple example is the spectrum of antiresonance for any driving-point frequency response function. In this spectrum the antiresonance frequencies correspond to the mode frequencies of the structure with the driving-point DOF constrained.

1. Simple Two Degree of Freedom Example

A simple two DOF example from (Gordis, 1999) will demonstrate this. The system is composed of masses, M1 and M2, and springs K1 and K2 and is undamped. This system is shown in Figure (3).

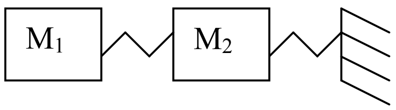


Figure 3. Two DOF system

From Eqn. (29) the driving point FRF is

$$H_{ii}(\Omega) = \sum_{k=1}^{modes} \frac{(\Phi_i^k)^2}{\omega_k^2 - \Omega^2} \tag{32}$$

Where ϕ_i^r is the mass normalized mode shape element, w_r, is the *r*th natural frequency, and Ω is the forcing frequency.

Solving for the antiresonance forcing frequency of $H_{11}(\Omega)$ leads to

$$\Omega^2_{anti-res} = \frac{R_{11}^1\omega_2^2 + R_{11}^2\omega_1^2}{R_{11}^1 + R_{11}^2} \tag{33}$$

Where the modal residual is given by $R_{ij}^r = \phi_i^r\phi_j^r$. Using values of $M_1 = M_2 = 1.0$, and $K_1 = K_2 = 1.0$. This generates mass and stiffness matrices

$$[K] = \begin{bmatrix} K_1 & -K_1 \\ -K_1 & K_1 + K_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}$$

$$[M] = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

yielding natural frequencies: $\{\omega(rad/\sec)\} = \begin{Bmatrix} 0.618 \\ 1.618 \end{Bmatrix}$ and a single anti-resonance located at the frequency $\Omega_{anti-res} = \sqrt{2}$ rad/sec. The driving point FRF, [H11] is shown in Figure 4. The single anti-resonance is noticeable at $\sqrt{2}$ rad/sec.

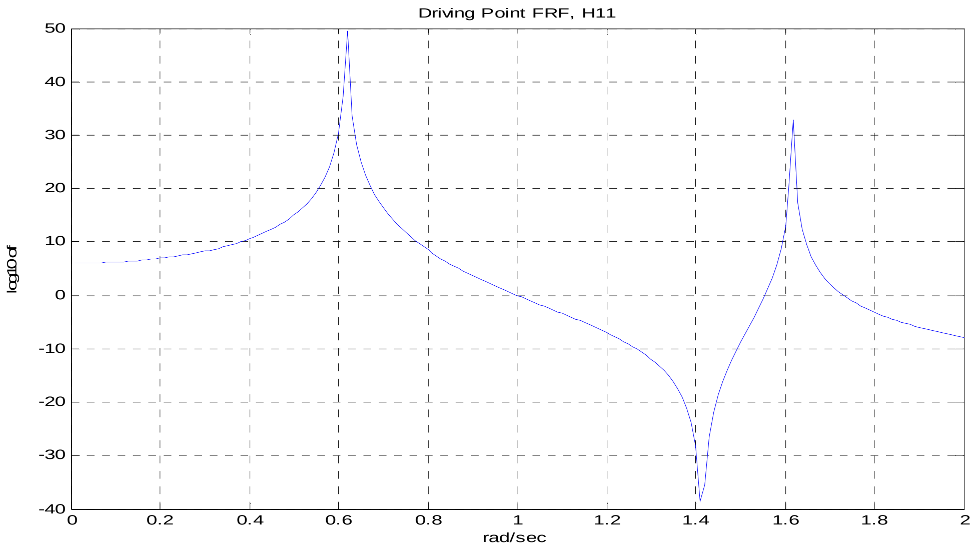


Figure 4. H_{11} Driving point FRF.

The same system is analyzed, however with DOF 1 constrained or pinned as shown in Figure (5).

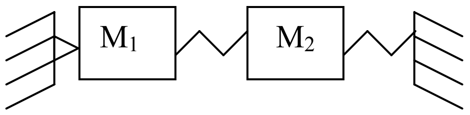


Figure 5. Two DOF system with ABC employed.

The resulting systems natural frequency lies at $\sqrt{2}$ rad/sec, which is identical to the original systems antiresonance frequency. This correlation is shown in Figure 6 below.

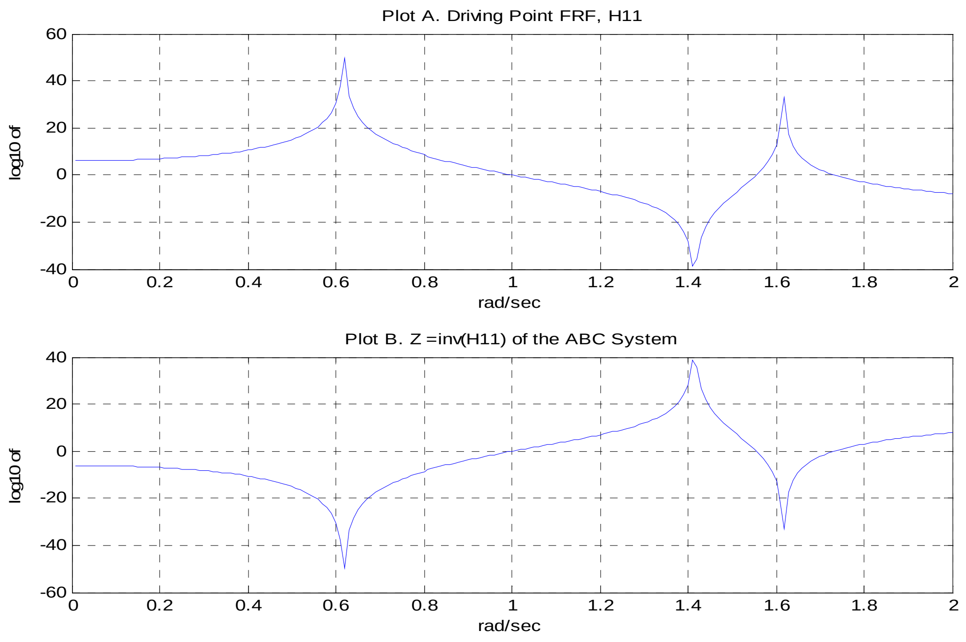


Figure 6. Plot A, Driving Point FRF of system 1, Plot B, Driving Point FRF of system 2.

2. ABC Frequencies for a Simply Supported Beam

The following example from (Fernandez, 2004) illustrates the use of ABC on a cantilever beam model. The model consists of ten elements, with two DOF per element yielding 20 total DOF.

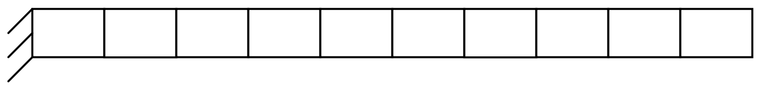


Figure 7. 10 element cantilever beam

Using Eqn. (29) the driving point FRF was calculated. The first eight natural frequencies of the system were calculated to be 4.9186, 30.826, 86.332, 169.29, 280.29, 419.91, 589.15, 789.30, all in HZ. The respective anti-resonances calculated are 6.8697, 43.856, 124.28, 245.53, 406.42, 586.39, 704.69, in Hz. The driving point FRF of the

$[H_{33}]$ was calculated using Eqn. (29) and plotted versus frequency. Next an ABC was the applied to the model at DOF 3 as shown in Figure 8.

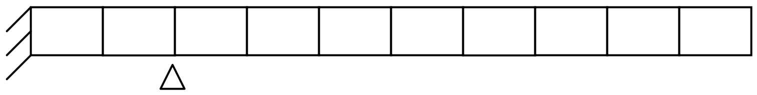


Figure 8. 10 element cantilever beam, ABC applied at DOF 3.

Again the natural frequencies were calculated, this time using the reduced order [K], [M] from the ABC system. The natural frequencies of the ABC system, DOF 3 pinned, under 800 Hz are 6.8697, 43.856, 124.28, 245.53, 406.42, 586.39, 704.69. Comparing these to the peaks of $[H_{33}]^{-1}$ a strong relationship can be seen. The same as example the plots of $[H_{33}]$ and $[H_{33}]^{-1}$ are combined on Figure (9) for easier comparison of the location of peaks and anti-resonances.

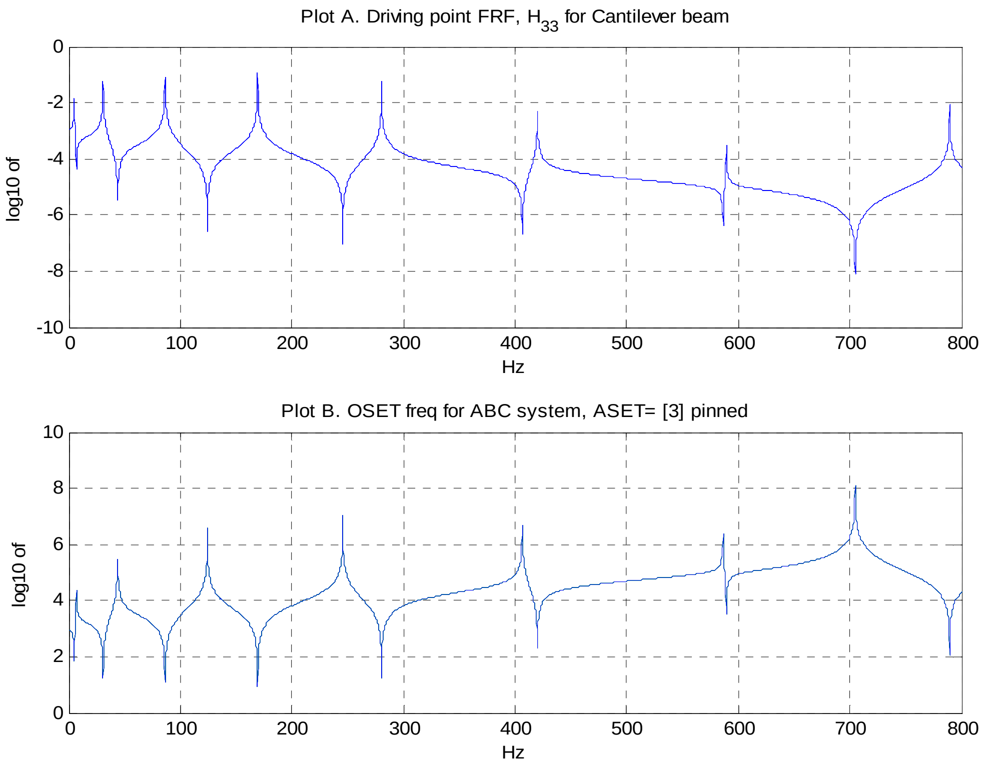


Figure 9. Plot A. Driving Point H_{33} (Ω) of free-free beam
Plot B. $[H_{33}(\Omega)]^{-1}$ with ABC applied at DOF 3

This demonstrates that the use of OSET natural frequencies can be used in improving and underdetermined problem solution. By making use of a just a single set of data from a modal parameter test of a structure, additional data can be extrapolated. This additional data is withdrawn from the OSET natural frequencies and easily applied to the model updating problem.

V. SENSITIVITY BASED UPDATING

As stated previously the matching of modal parameters between a FEM and the structure it represents is of great significance in structural dynamics. In order to do this the model often must be updated to obtain the actual modal parameters of the structure. When updating a finite element model to more closely match the actual structure the method of sensitivity based updating is often employed. The governing equation for sensitivity based updating is

$$\left\{\Delta w^2\right\} = \left[T\right]\left\{\Delta Dv\right\} \tag{34}$$

where $\left\{\Delta w\right\}$ is the change in natural frequency, $\left\{\Delta Dv\right\}$ is the change in design variable to be solved for, and $\left[T\right]$ is the sensitivity matrix, composed of first-order sensitivities.

A. SENSITIVITY MATRIX

The sensitivity matrix can be defined as how a change in a model parameter for one element affects the other elements of the model. This change could be a change of element mass or stiffness and its respective change on the systems change in natural frequency. In matrix form:

$$\left[T\right] = \left[\frac{\partial \omega_n^2}{\partial Dv}\right] \tag{35}$$

where each column represents an element of the model and each row represents a mode of the model.

The programs used to compose the sensitivity matrix for this study did so doing the following steps:

1. A small perturbation (1%) of mass is applied to the beam model on element 1.
2. The mass matrix is assembled for the mass perturbation, then the mass matrix and partial derivative are calculated.

3. The first column of the sensitivity matrix is calculated using:

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\{\Phi_i\}^T\left[\frac{\Delta K}{\Delta DV}-\lambda_i\frac{\Delta M}{\Delta DV}\right]\{\Phi_i\} \tag{36}$$

Note: This equation is part of derivation of the sensitivity matrix that is presented in the next section.

4. The process is now repeated for element 2, and continues to be completed for all elements.

5. Once the mass portion has been calculated the same process is repeated with respect to the stiffness of each element. Again a 1% perturbation was used.

6. The two separate portions (mass and stiffness) are then combined to form the total sensitivity matrix shown below. For this study only mass and stiffness varied, however many other design variables can be used in the sensitivity matrix.

$$[T]=\begin{bmatrix}\frac{\partial w_1^2}{\partial Dv_1}\cdots\frac{\partial w_1^2}{\partial Dv_2}\\ \vdots\ddots\vdots\\ \frac{\partial w_n^2}{\partial Dv_1}\cdots\frac{\partial w_n^2}{\partial Dv_2}\end{bmatrix} \tag{37}$$

This is considered the base system sensitivity matrix because it does not take into account any artificial boundary conditions. When adding ABC the process is repeated, but the mode shapes used are those of the ABC system. The total system is described below including *n* number of ABC.

$$[T]=\begin{bmatrix}\begin{bmatrix}\dfrac{\partial w_1^2}{\partial Dv_1}\cdots\dfrac{\partial w_1^2}{\partial Dv_2}\\ \vdots\ddots\vdots\\ \dfrac{\partial w_n^2}{\partial Dv_1}\cdots\dfrac{\partial w_n^2}{\partial Dv_2}\end{bmatrix}\\ \begin{bmatrix}\dfrac{\partial w_1^2}{\partial Dv_1}\cdots\dfrac{\partial w_1^2}{\partial Dv_2}\\ \vdots\ddots\vdots\\ \dfrac{\partial w_n^2}{\partial Dv_1}\cdots\dfrac{\partial w_n^2}{\partial Dv_2}\end{bmatrix}\\ \vdots\\ \begin{bmatrix}\dfrac{\partial w_1^2}{\partial Dv_1}\cdots\dfrac{\partial w_1^2}{\partial Dv_2}\\ \vdots\ddots\vdots\\ \dfrac{\partial w_n^2}{\partial Dv_1}\cdots\dfrac{\partial w_n^2}{\partial Dv_2}\end{bmatrix}\end{bmatrix} \tag{38}$$

This shows that a change in model parameters at one element will alter the modal parameters of the entire model. However, the amount of change in modal parameters is dictated by the sensitivity matrix, allowing a perturbation to have significant or minimal effect. It is because of the influence that the sensitivity matrix is a key component in determining error prediction in a finite element model. Several properties of the sensitivity matrix will be investigated later to understand its' ability in error detection and prediction

B. DERIVATION OF SENSITIVITY MATRIX

To generate the sensitivity matrix, starting with the eigenvalue problem of a conservative *n* DOF

$$[K-\lambda_i M]\{\Phi_i\}=\{0\} \tag{39}$$

Now taking Eqn. (39) and taking the derivative with respect to the design variable generates Eqn. (40).

$$\left[\frac{\Delta K}{\Delta DV}-\lambda_i\frac{\Delta M}{\Delta DV}-\frac{\Delta\lambda_i}{\Delta DV}\right]\{\Phi_i\}+\left[K-\lambda_i M\right]\left\{\frac{\Delta\Phi_i}{\Delta DV}\right\}=\{0\} \qquad (40)$$

Expanding Eqn. (40) and pre-multiplying by $\{\Phi\}^T$ leads to Eqn. (41)

$$\{\Phi_i\}^T\left[\frac{\Delta K}{\Delta DV}\right]\{\Phi_i\}-\lambda_i\{\Phi_i\}^T\left[\frac{\Delta M}{\Delta DV}\right]\{\Phi_i\}-\left[\frac{\Delta\lambda_i}{\Delta DV}\right]\{\Phi_i\}^T[M]\{\Phi_i\}+\{\Phi_i\}^T[K-\lambda_i M]\left\{\frac{\Delta\Phi_i}{\Delta DV}\right\}=\{0\} \qquad (41)$$

Now invoking the use of the matrix identity $\{a\}^T[b]\{c\}=\{c\}^T[b]\{a\}$ the last term on the left hand side can be replaced by Eqn. (42), which applying Eqn. (39) will lead to Eqn. (43)

$$\left\{\frac{\Delta\Phi_i}{\Delta DV}\right\}^T\left[K-\lambda_i M\right]\{\Phi_i\}=0 \qquad (42)$$

$$\left\{\frac{\Delta\Phi_i}{\Delta DV}\right\}^T\left[K-\lambda_i M\right]\{\Phi_i\}=0 \qquad (43)$$

This then reduces Eqn. (41) to

$$\{\Phi_i\}^T\left[\frac{\Delta K}{\Delta DV}\right]\{\Phi_i\}-\lambda_i\{\Phi_i\}^T\left[\frac{\Delta M}{\Delta DV}\right]\{\Phi_i\}-\left[\frac{\Delta\lambda_i}{\Delta DV}\right]\{\Phi_i\}^T[M]\{\Phi_i\}=\{0\} \qquad (44)$$

Now using orthogonality, where $[\Phi_i]^T[M][\Phi_i]=1$ Eqn. (44) further reduces to

$$\{\Phi_i\}^T\left[\frac{\Delta K}{\Delta DV}\right]\{\Phi_i\}-\lambda_i\{\Phi_i\}^T\left[\frac{\Delta M}{\Delta DV}\right]\{\Phi_i\}-\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\{0\} \qquad (45)$$

Now bringing the mass and stiffness portion to the right hand side hand side

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right]=\{\Phi_i\}^T\left[\frac{\Delta K}{\Delta DV}-\lambda_i\frac{\Delta M}{\Delta DV}\right]\{\Phi_i\} \qquad (46)$$

This can be broken down further into its separate parts consisting of the mass portion and stiffness portion.

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right] = \{\Phi_i\}^T \left[\frac{\Delta K}{\Delta DV}\right]\{\Phi_i\} \quad \text{where} \quad [\Delta K] = [K_x - K_a] \tag{47}$$

$$\left[\frac{\Delta\lambda_i}{\Delta DV}\right] = \{\Phi_i\}^T \left[-\lambda_i \frac{\Delta M}{\Delta DV}\right]\{\Phi_i\} \quad \text{where } [\Delta M] = [M_x - M_a] \tag{48}$$

For both the mass and stiffness the influence of the mode shapes is evident. Normalizing each row of the mass and stiffness sensitivity matrix demonstrates this. Figures (10) and (11) show the base system mass and stiffness sensitivity matrix rows normalized for the first five rows. These first five rows correspond to the first five modes of the base system. Each element is represented by a bar indicating its influence on the change of the systems natural frequency with a respective change of design variable. For instance, the stiffness sensitivity matrix indicates that a change in stiffness closer to the clamped end of the beam would have a much greater influence on the change of natural frequency of the system as opposed to a change of stiffness at the free end of the beam. Just the opposite is true for the mass sensitivity matrix. It should be noted that the mass sensitivity rows are an absolute value and that an increase of mass would generally lower a systems naturally frequency, however, it was desired to show the influence with respect to element position and thus all magnitudes were made positive.

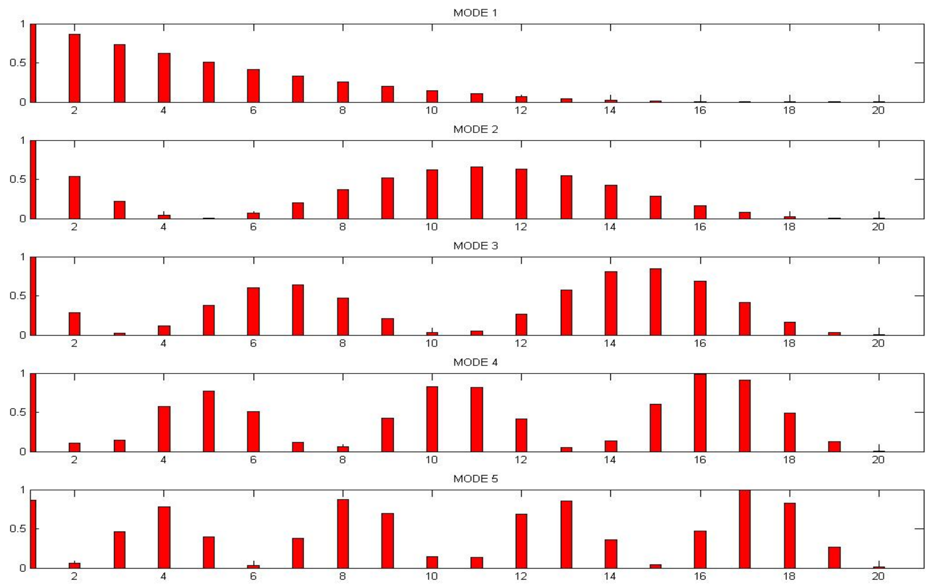


Figure 10. Base system, stiffness sensitivity matrix, modes 1:5

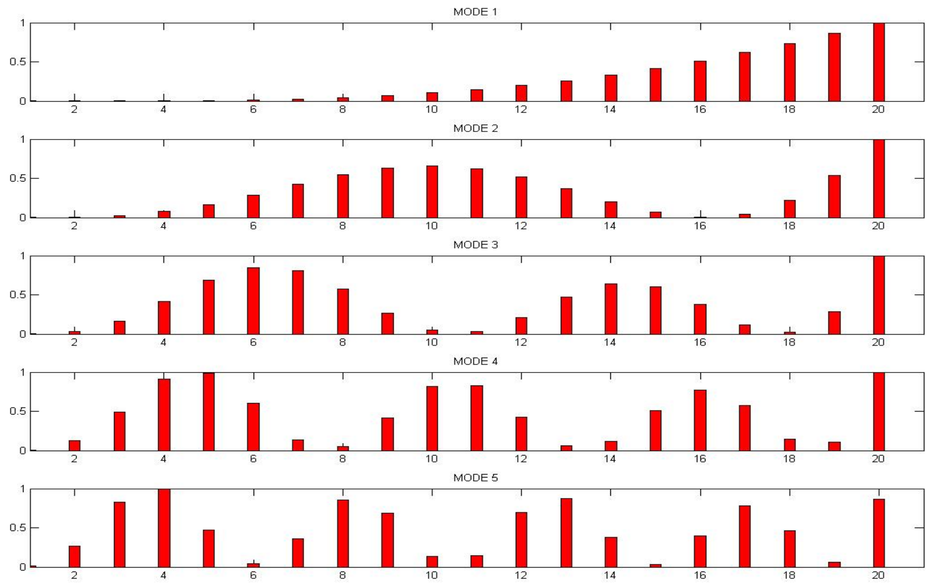


Figure 11. Base system, mass sensitivity matrix, modes 1:5

From these figures it can be seen that where the graphs show elements of low sensitivity correspond to areas where the sensitivity matrix would not identify a change in design variable. It should be noted that due to the normalization of each row the graphs are misleading in the frequency influence. The higher modes have a much larger change of frequency with a change in design variable. In other words a change in the design variable of row one at the element with magnitude of one would produce a much smaller change in frequency than a change in the design variable at the element with magnitude of one in row two and so on.

Figures (12) and (13) show how the application of an ABC alters the respective rows the sensitivity matrix. The green triangle indicates the DOF where the ABC has been applied to the system. In general for stiffness the application of an ABC drives up the values of the sensitivity matrix in that region where as for mass it pushes the larger values away from the restrained DOF. Again this was intuitive from the resulting mode shapes when the ABC are applied. Figure (14) plots the modes for both the base and ABC system. The effect of the ABC on the mode shapes corresponds to that of the sensitivity matrix.

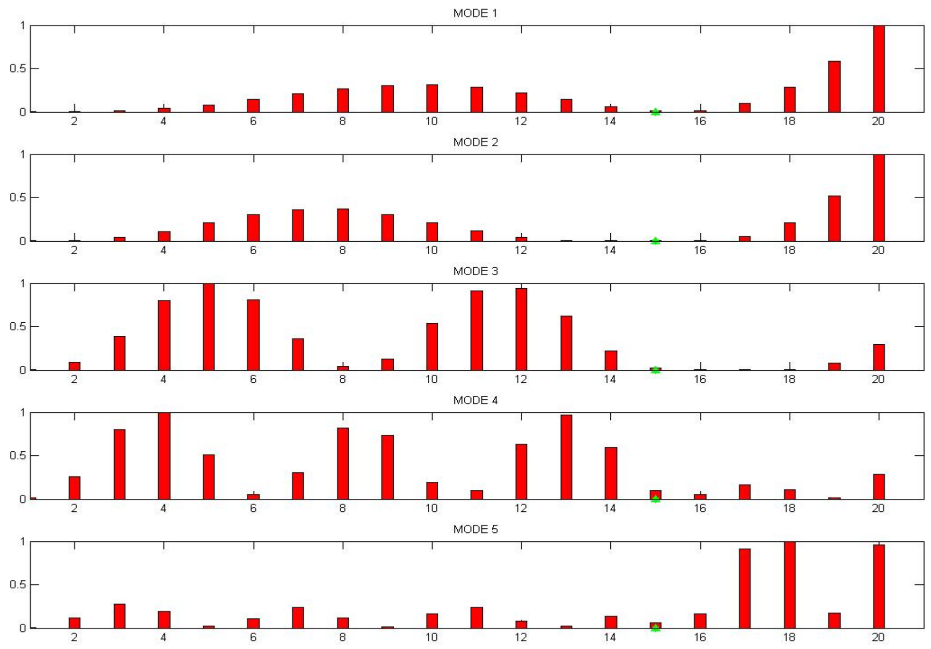


Figure 12. ABC system, DOF 31 pinned, mass sensitivity matrix, modes 1:5

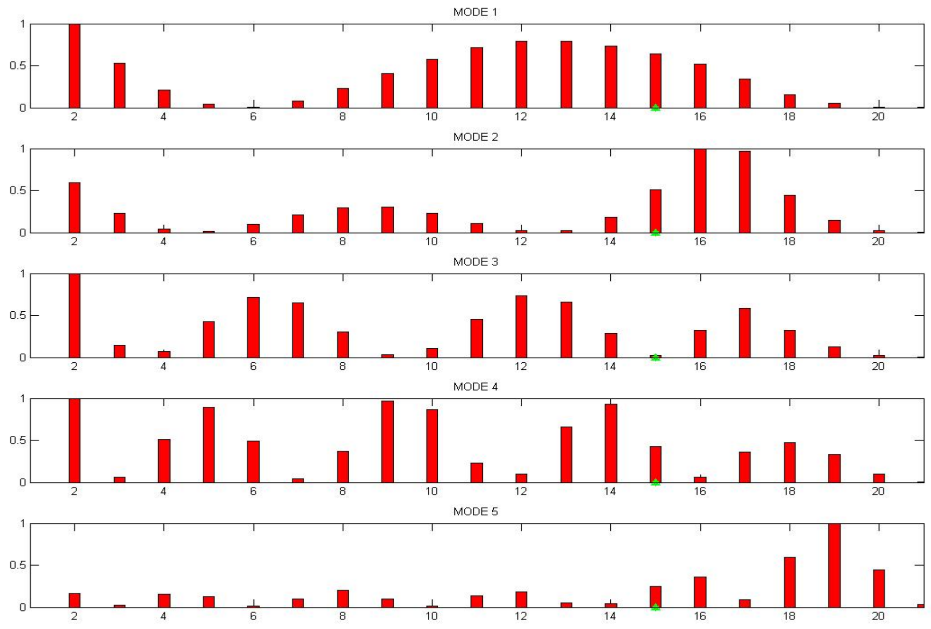
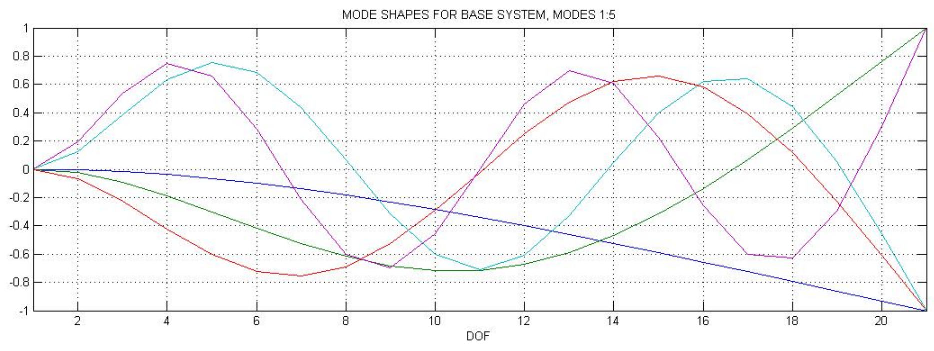


Figure 13. ABC system, DOF 31 pinned, stiffness sensitivity matrix, modes 1:5



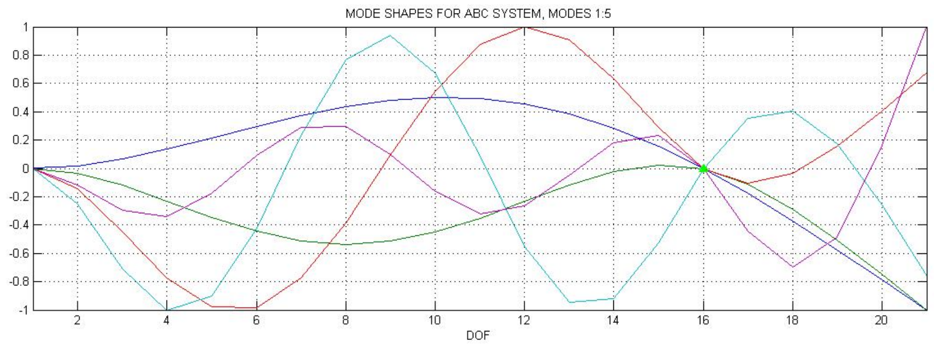


Figure 14. Plot A. Base System, Modes 1:5. Plot B. ABC System DOF 31 pinned , Modes 1:5

The relationship between mode shapes and the sensitivity matrix often plays a key role in the updating of a FEM. The use of ABC can play a key role in helping to produce a well conditioned problem and solution in the updating of the model. As discussed before a poorly conditioned problem will be one that has a poor solution. However, a well conditioned problem, while it is not guaranteed to have a quality solution it is not ruled out by conditioning.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. ERROR PREDICTION

An analysis of error prediction was conducted using MATLAB to create two separate beams. Each beam was identical in dimensions, 42 inches, 1.5 inches, 1inch and equal density and Young's Modulus. Each beam was modeled with a twenty element F.E.M. However, one beam was considered the analytical beam, which represented the baseline beam, or the F.E.M. The second beam was considered the experimental beam that had a known perturbation applied at a known element. Although, this beam too was a F.E.M it represented an experimental structure from the laboratory. The errors imposed on the experimental beam consisted of mass or stiffness errors, generally in the amount of 10 percent. A diagram of the cantilever beam is shown below in Figure (15).

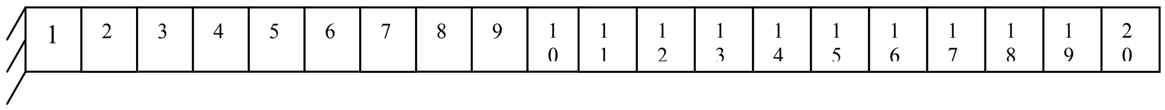


Figure 15. Cantilever Beam

Numerous trials were run with the program to help identify error prediction and any correlations or existing indicators of a strong or weak solution. Then for the base and all ABC error prediction was computed. This was done for summations of modes one through twenty. For each of these runs the condition of the sensitivity matrix was computed as well as the rank. One of the areas of interested was the number of modes required to attain a solution within ten percent of the actual error at the affected element.

To solve for the error the matrix equation $\{\Delta w^2\} = [T]\{\Delta Dv\}$ was used. In order to solve for the change in design variable $\{\Delta Dv\}$ MATLAB arranged the equation as follows

$$\{\Delta Dv\} = [T]^{-1}\{\Delta w^2\} \tag{49}$$

this is a challenging problem due to the nature of $[T]$ which is not symmetric when dealing with less than 20 modes in the solution. Due to this several different solution methods were investigated to try and develop better performance from MATLAB. In order to fully understand the results from the MATLAB program a brief discussion of the solution methods can give insight as to why certain values were obtained or not obtained.

A. UNDERDETERMINED MATRIX ALGEBRA SOLUTIONS

Looking at the matrix problem is standard form

$$[A]\{x\}=\{b\} \tag{50}$$

where $[A]$ represents the sensitivity matrix, $\{b\}$ is the square of the change in natural frequency, and $\{x\}$ is the "error" or difference in stiffness or mass between the experimental and analytical model. Or more completely

$$\begin{bmatrix} A_{11} \cdots A_{1n} \\ \vdots \ddots \vdots \\ A_{m1} \cdots A_{mn} \end{bmatrix} \begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_m \end{Bmatrix} \tag{51}$$

Where *n* columns represents the elements of the model and *m* rows represent the number of modes retained in the sensitivity matrix. The $\{x\}$ vector is always of *n* length and the $\{b\}$ vector is of length *m*. With the particular problem being addressed $m < n$ thus there are several methods to solve the problem.

1. Condition and Rank of a Matrix

Of great importance in the matrix solution of a problem is the "health" of the matrices involved. Some of the most important indicators of the health of a matrix are its rank and condition. Matrix rank can be thought of as the common dimension of the row space and column space of that given matrix (Anton, 2005). Essentially this is saying the rank is the number of linearly independent rows that exist in a matrix. Thus if a *n* by *m* matrix has a rank less than *n* it is considered singular and all of its rows are not linearly independent resulting in fewer solution equations.

The condition number of a matrix is a measure of the sensitivity of the linear system $[A]\{x\}=\{b\}$. Essentially this is a measure of the change in the solution ("x") with a small change in either "A" or "b" or both. A healthy matrix would have a condition number close to one, where as a very large condition number would indicate a matrix that would not lead to a reliable solution (Watkins, 2002). A good condition number can lead to a reliable solution depending on the solution method that was utilized. Several solution methods are discussed next.

2. Pseudoinverse Method

The inverse of $[A]$ exists only if $[A]$ is square and has full rank. If this is the case the solution is

$$\{x\}=[A]^{-1}\{b\} \tag{52}$$

The pseudoinverse, $[A]^{+}$ is a generalization of the inverse and exists for any matrix and the solution to $[A]\{x\}=\{b\}$ becomes

$$\{x\}=\left[A\ \right]^{+}\{b\} \tag{53}$$

The best way to generate $[A]^{+}$ is by use of singular value decomposition which provides a numerically robust solution to the least squares problem.

$$A=USV^{T} \tag{54}$$

Where U and V are orthogonal and are (*n*,*n*) and S is diagonal of dimensions (*m*,*n*) with real, non-negative singular values. This leads to

$$A^{+}=V(S^{T}S)^{-1}S^{T}U^{T} \tag{55}$$

However, in the under-determined case because the rank, r, of $[A]$ is less than n, $\left(A^T * A\right)^{-1}$ does not exist, thus the program uses only the first r singular values reducing S to an (r,r) matrix and shrinking U and V accordingly (Bock, 1998). Combining Eqn. (54) and (55) leads to

$$x = VS^{-1}U^T b \tag{56}$$

When this solution method was incorporated in the error solution problem it gave results that tended to distribute the error location across the beam rather than look for a point mass or stiffness solution. As a result the error predicted at the affected element was well below the actual value. This is shown in Figure (16) below. The "actual" error is .1 located at element 15, the Pseudoinverse method distributes the error across all elements. Thus, for this particular application, this method was not seen as a reliable solution.

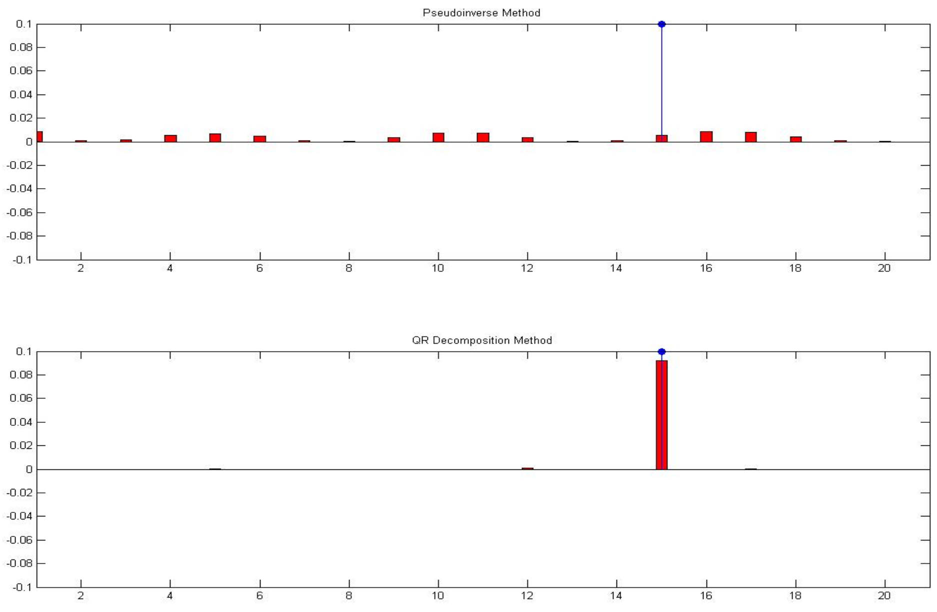


Figure 16. Pseudoinverse Method vs. QR Decomposition Method

3. Orthogonal-triangular (QR) Decomposition

Orthogonal-triangular decomposition or QR decomposition expresses the matrix as the product of a real complex unitary matrix, Q and an upper triangular matrix, R. The Householder method of this type allows for column pivoting. Thus, as the program searches for a solution it will select columns of $[A]$ that are more orthogonal to the other used columns and push less orthogonal columns further down in the equation. This does produce fairly accurate results with the number of non-zero scalars in the $\{x\}$ vector equal to the rank of $[A]$. This gives results that do not allow for the examination of some of the properties of $[A]$ due to the change in columns. By using a QR method that does not allow for column pivoting, the properties of $[A]$ are retained and can be examined. In particular the condition of the sensitivity matrix was examined. The use of column pivoting did improve the results in terms of number of modes needed to locate the error. Table (1) and (2) below show this. Table (1) shows the results with the use of column pivoting and indicates that the error is found with 6 modes, where as Table (2) shows the results without employing column pivoting and it takes 9 modes to find the error.

Modes	***1***	***1:2***	***1:3***	***1:4***	***1:5***	***1:6***	***1:7***	***1:8***	***1:9***	***1:10***
Element	Perceived Error (percent)									
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0.0126	0	0.006	0.0186
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0.0352	0.0155	0.0221	-0.001	0.0066	-0.004	0	0.012
6	0	0	0	0	0	0	0	0.01	0.0059	-0.004
7	0	-0.008	0	0	-0.012	0	0	0	0	-0.002
8	0	0	-0.022	-0.008	0	0	-0.002	-0.005	-0.004	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0.0054	0.0002	-0.011	-0.002	-0.004	-0.008
12	0	0	0	0	0	0	0	0.0044	-0.001	-0.012
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0.1015	0.0924	0.0946	0.0965	0.102
16	0	0	0	0.0387	0	0	0	0	-0.003	-0.013
17	0	0	0	0	0.0374	-0.001	-0.001	0.0043	0.0035	0.004
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
20	0.0034	0.0035	0.0039	-0.006	-0.005	0.0003	0.001	-0.008	-0.000	0.0009

Table 1. QR Decomposition with column pivoting

Modes	***1***	***1:2***	***1:3***	***1:4***	***1:5***	***1:6***	***1:7***	***1:8***	***1:9***	***1:10***
Element	Perceived Error (percent)									
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	-0.001	0.0001
3	0	0	0	0	0	0.0478	0.0443	0.0599	0	0.0187
4	0	0	0	0	0.0228	0	-0.021	-0.037	-0.001	-0.053
5	0	0	0	0.0425	0	0.0031	0	0	0.0005	0
6	0	0	0.0165	0	0.026	0	0.0558	0.0423	0	0.0942
7	0	0	0	0	0	0.0388	0	0	0	0
8	0	0	0	0	0	0	-0.053	0	0	0
9	0	0	0	0	0	0	-0.005	0	0	0
10	0	0	-0.069	0	-0.062	-0.068	-0.034	-0.035	0	-0.012
11	0	0	0	-0.047	0	0	0	0	0	0
12	0	0	0	0	0	0	0	-0.019	-0.000	-0.012
13	0	0	0	0	-0.081	-0.12	0	-0.050	-0.001	-0.011
14	0	0	0	0	0	0	0	0	0.0014	-0.007
15	0	0	0	0	0	0	0	0	0.0972	0
16	0	0	0	0	0	0	0	0.0267	0	0.0075
17	0	0.0603	0	0.0328	0	0	0	0	0.0008	0.0737
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
20	0.0415	0.0042	0.0485	0.0278	0.0687	0.0809	0.0471	0.048	0	0

Table 2. QR Decomposition without column pivoting

The nature of this method proved to give results that reflected the point mass or stiffness at a particular element rather than distributing the mass or stiffness across the entire length of the beam like the Pseudoinverse method did. Once this method found the error at the correct location it was usually within ten percent of the actual error. Of concern with this method was that it would frequently pick up the error in the correct location and amount, retain the solution for an additional two or three modes, then lose the solution completely at the location as an additional few modes were added. Then the solution was regained with the addition of more modes. The reason for this "losing" of the solution lies in how the program solves the underdetermined problem and looks for the solution that minimizes the length of the vector $[A]\{x\}-\{b\}$. Thus, as the program was searching for the "shortest" solution to the problem at times that solution would not include the desired element error. This can be seen in Table (3) where the error in element 15 is identified after 4 modes, but then lost as modes 5 and 6 are incorporated, then found again with mode 7 added. Then the error was lost for mode 8 then identified again as mode 9 was incorporated. It was attempted to find a predictor that would indicate when the solution would drop out the correct location. It was thought that a dramatic increase in condition or a decrease in rank might indicate when the error would be dropped, but these did not turn out to be indicative of when the error would be dropped. The change in condition can be shown from Table (3) where from modes 4 though 9 the error is found then lost several times. Throughout the process the condition number increases but not more dramatically when the error is lost or regained.

Tables (3) and (4) compare the results of the Orthogonal-Triangular Decomposition and the Pseudoinverse method. The example is a stiffness error at element 15, with an ABC applied at DOF 9. Element 15 is highlighted in red and elements 4 and 5 in blue to indicate that their shared DOF is the DOF with the applied ABC. Modes 1 through 10 were summed and the condition of the respective sensitivity matrix was taken for each summation. The table clearly shows how the Orthogonal-Triangular Decomposition method finds the error at the sum of modes 1 through 4, then loses it at as modes 5 and 6 are applied. The table also shows how the Pseudoinverse method spreads out the error and does not perceive a point error.

Modes	***1***	***1:2***	***1:3***	***1:4***	***1:5***	***1:6***	***1:7***	***1:8***	***1:9***	***1:10***
Condition	1	42.740	347.26	1361.1	3707.6	7673.3	13356	20088	35573	58975
Element	Perceived Error (percent)									
1	0	0	0	0	0	0	0.001	0.0501	0.0002	0.0006
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0.087	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0.003	0.0032	0.0023	0.0001	0.0015	0.0176	0.0001	0	0	0
6	0	0	0	0	0	0	0	0.0527	0.0002	0
7	0	0	0	0	0	0	0	0.0694	0	0.0019
8	0	0	0	0	0	0	0	0	0	0.0019
9	0	0	0	0	0	0	0	0	0	0.0005
10	0	0	0	0	0	0	0.0007	0	0.0007	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0.0261	0.0007	0.0223	0.0218	0.0007	0.0263	0.0002	0.0001
13	0	0.0626	0	0	0	0	0	0	0	0
14	0	0	0	0	0.0682	0.0622	0	0.047	0.0012	0.0022
15	0	0	0	0.0922	0	0	0.0926	0	0.0928	0.0946
16	0	0	0.0878	0	0.0555	0.056	0	0.0586	0.0001	0.0007
17	0	0	0	0.0006	0	0	0.0005	0	0	0
18	0	0	0	0	0.0222	0.0378	0	0.0704	0.0001	0.0008
19	0	0	0	0	0	0	0	0.1375	0.0008	0.002
20	0	0	0	0	0	0	0	0	0	0

Table 3. Orthogonal-triangular decomposition

Modes	***1***	***1:2***	***1:3***	***1:4***	***1:5***	***1:6***	***1:7***	***1:8***	***1:9***	***1:10***
Condition	1	42.740	347.26	1361.1	3707.6	7673.3	13356	20088	35573	58975
Element	Perceived Error (percent)									
1	0.0004	0.0073	0.0166	0.0088	0.0005	0.0053	0.0082	0.0029	0.0014	0.0053
2	0.0003	0.0039	0.0047	0.0009	0	0.001	0.0038	0.0024	0.0017	0.0076
3	0.0003	0.0016	0.0003	0.0013	0.0003	0.0052	0.0097	0.0029	0.0009	0.002
4	0.0002	0.0003	0.0019	0.0051	0.0005	0.0037	0.0021	0.0007	0.0013	0.0088
5	0.0002	0	0.0063	0.0068	0.0002	0.0004	0.0051	0.0044	0.0019	0.0028
6	0.0002	0.0005	0.01	0.0045	0	0.0037	0.0113	0.0018	0.0005	0.008
7	0.0001	0.0015	0.0105	0.001	0.0002	0.0064	0.0029	0.0018	0.0024	0.0036
8	0.0001	0.0027	0.0078	0.0005	0.0005	0.0023	0.0042	0.0043	0.0005	0.007
9	0.0001	0.0038	0.0035	0.0038	0.0004	0.0009	0.0114	0.0007	0.002	0.0047
10	0.0001	0.0045	0.0005	0.0073	0.0001	0.0054	0.0035	0.0033	0.0012	0.0058
11	0	0.0048	0.0008	0.0072	0.0001	0.0054	0.0035	0.0033	0.0011	0.0059
12	0	0.0046	0.0044	0.0036	0.0004	0.0009	0.0114	0.0007	0.002	0.0047
13	0	0.004	0.0096	0.0005	0.0005	0.0023	0.0042	0.0043	0.0005	0.007
14	0	0.0031	0.0134	0.0012	0.0002	0.0063	0.0028	0.0018	0.0024	0.0036
15	0	0.0021	0.0141	0.0053	0	0.0036	0.0112	0.0017	0.0005	0.008
16	0	0.0012	0.0114	0.0087	0.0003	0.0004	0.0049	0.0043	0.0019	0.0027
17	0	0.0006	0.0068	0.008	0.0006	0.0045	0.0025	0.0007	0.0013	0.0086
18	0	0.0002	0.0028	0.0043	0.0005	0.0073	0.0124	0.0036	0.0011	0.0023
19	0	0	0.0006	0.0011	0.0002	0.0033	0.0085	0.0043	0.0027	0.0109
20	0	0	0	0.0001	0	0.0002	0.0007	0.0004	0.0004	0.002

Table 4. Pseudoinverse Method

B. ERROR PREDICTION SOLUTION

Numerous trials were run to help identify error prediction and any correlations or existing indicators of a strong or weak solution. For each element a perturbation of mass or stiffness of ten percent was applied. Then for the base and all ABC error prediction was computed. This was done for summations of modes one through twenty. For each of these runs the condition of the sensitivity matrix was computed as well as the rank. The rank for every run was identical to the number of modes retained and thus was not seen as a good predictor of solution accuracy. The condition number was greater for the systems that required more modes to achieve the error tolerance. But again this was not seen as a good predictor because the condition of the sensitivity matrix is independent of error location.

The following example demonstrates the process and some of the key components of the results. Using the twenty element beam model element 15 was subjected to the ten percent mass or stiffness perturbation. The system was modeled under the base condition (no ABC), ABC with DOF 9 pinned then ABC with DOF 31 pinned. These conditions are displayed in Figure (17).

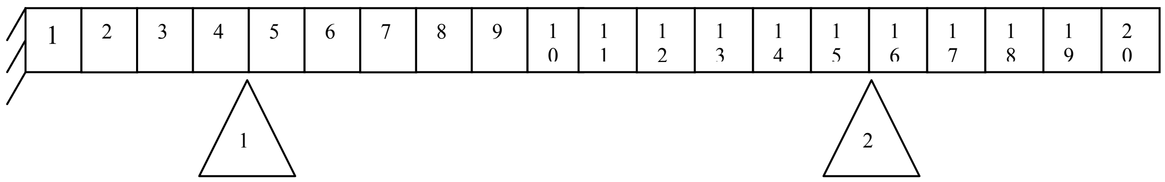


Figure 17. Twenty element beam, ABC system with DOF 9 pinned (1) and ABC system with DOF 31 pinned (2).

Figures (18) show the results for the applied error. In the graph the red bars indicate the base system, blue indicates the system with DOF 9 pinned, and green with DOF 31 pinned. The stem plot indicates the magnitude and where the error or damage is located.

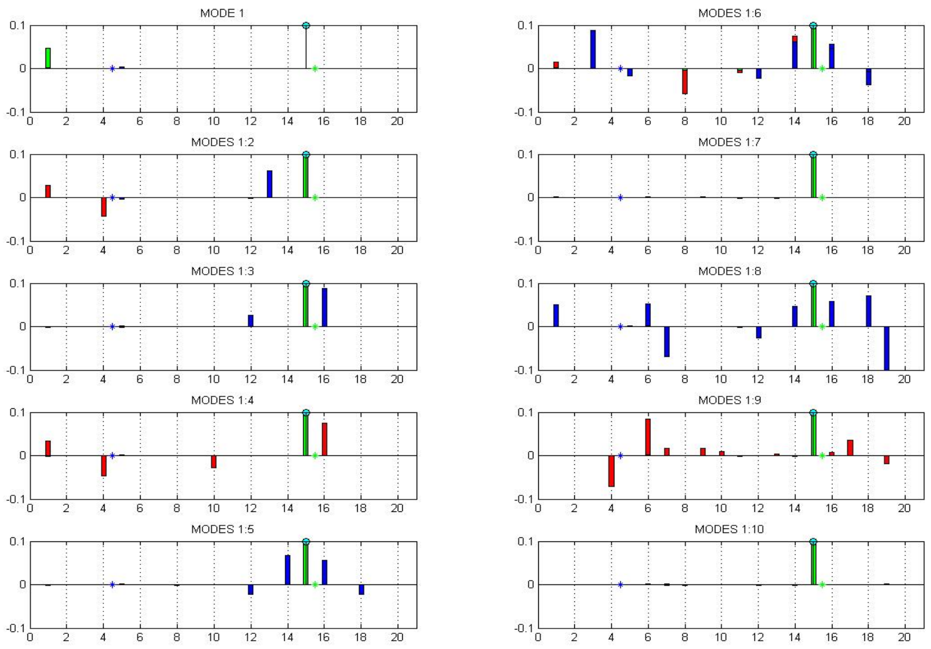


Figure 18. Element 15, Stiffness

The stiffness perturbation applied to element 15 was difficult for the base system to pick out due to its location in a region of the beam that was relatively insensitive to a change in stiffness. The two ABC did have better success in detecting the error. In particular the ABC applied at DOF 31 picked up the error quickly and retained it. This demonstrates one of the key findings in how when trying to predict the error location the model was highly influenced by areas of strong sensitivity. Errors in those areas were detected with relative ease. In particular with stiffness applying an ABC adjacent to the element with error can detect and keep the error location very well. This was a reflection of the increased sensitivity when applying boundary conditions which had the effect of increasing the sensitivity in the general region. This can be seen in Figure (19) below where the top graph is the base system, the second ABC with DOF 9 pinned, the third ABC with DOF 19 pinned and fourth ABC with DOF 29 pinned. The one area where this was not particularly true was at the free end of the beam where the application of an ABC did not dramatically increase the sensitivity. This can be seen as a result of the strain energy at the end of a pinned beam being relatively small thus a change in stiffness at this point would not have an strong influence on the natural frequency of the system.

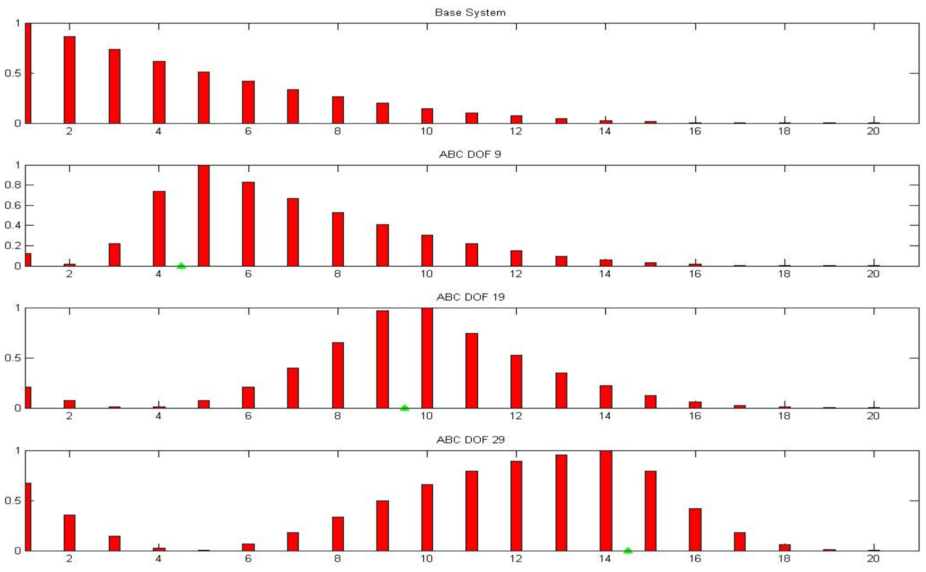


Figure 19. Movement of pinned DOF effect on sensitivity

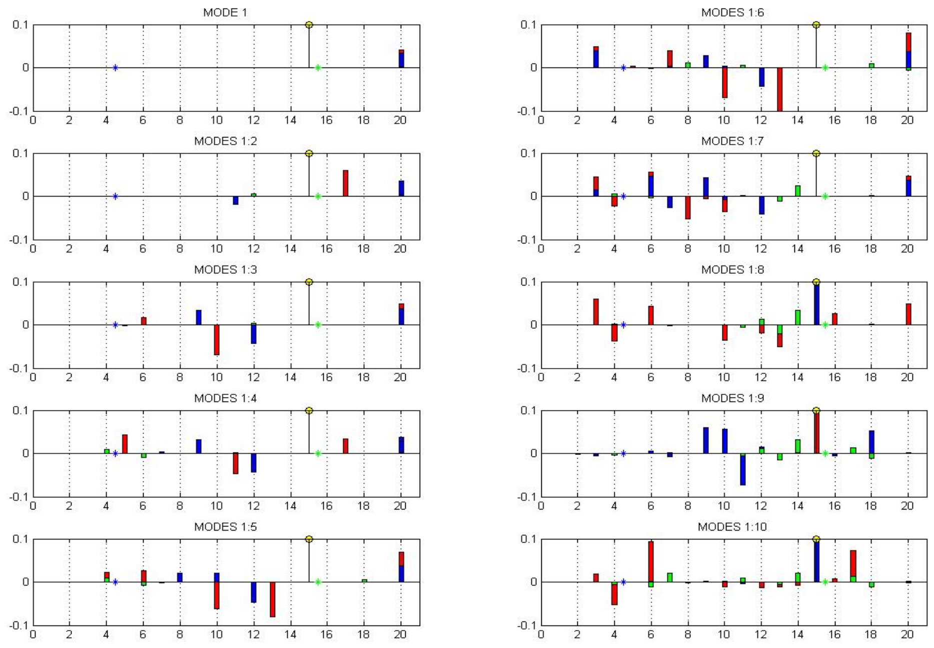


Figure 20. Element 15, Mass

When dealing with a change of mass the system tended to have similar behavior but it was somewhat more difficult to anticipate. It was expected that the base system and the ABC system with DOF 9 pinned would have success locating the mass error at element 15. This was only true once 8 or more modes were retained. Again the system performed better with areas of high sensitivity finding the damage in the structure. This was however, more difficult to anticipate with the mass perturbation due to its behavior with respect to additions of boundary conditions.

Next an investigation of error prediction was done with the combination of base and ABC modes. Again element 15 was used, with ABC at DOF 9 and 31. The first three modes for both the base system and ABC system were summed for each of the three systems mass and stiffness perturbations. It should be noted that only one ABC at a time was combined with the base system Figures (21) and (25) display the results. The stem plot indicates the magnitude and location of the error. The base system in combination with the ABC system with DOF 9 pinned is displayed in blue. The base system in combination with the ABC system with DOF 31 pinned is displayed in green.

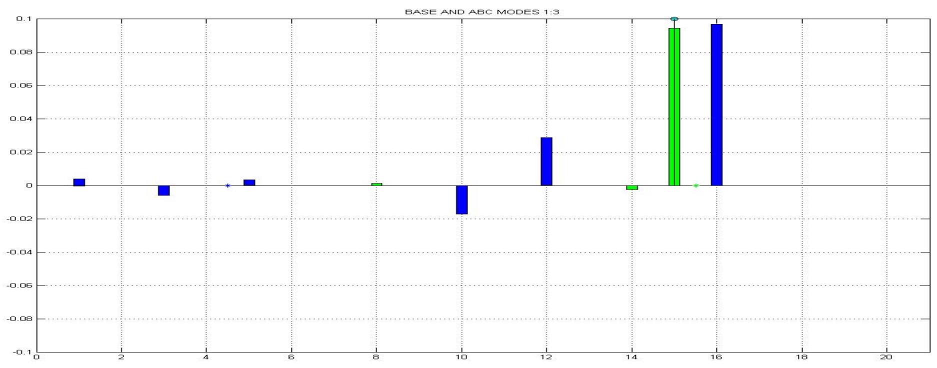


Figure 21. Element 15, Base and ABC systems, Modes 1:3, Stiffness

When dealing with the combination of two boundary conditions on a system the performance was based on how the boundary conditions complimented each other on fortifying areas of low sensitivity. The smoother the sensitivity of the system, (fewer areas of extremely low sensitivity) the stronger the results. This can be thought of as combining sensitivities of different boundary conditions to build up areas of low sensitivity to help find the error. If the error exists in an area of low sensitivity then it will be difficult to locate, thus by eradicating areas of low sensitivity error location is improved. Figures (22) through (24) show the sensitivities for the base system, ABC with DOF 9 pinned and ABC with DOF 31 pinned. The pinning of both DOF 9 and DOF 31 creates sensitivities in areas that the base system is relatively insensitive. As a result the error is detection is generally improved.

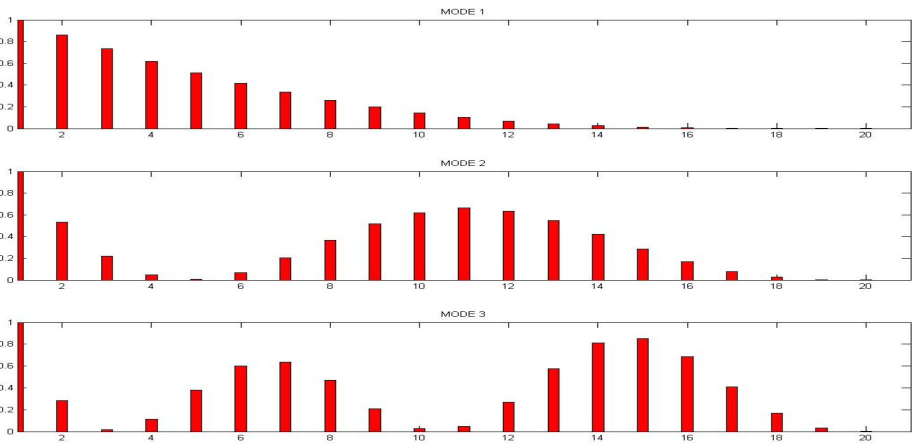


Figure 22. Base System, Modes 1-3, Stiffness

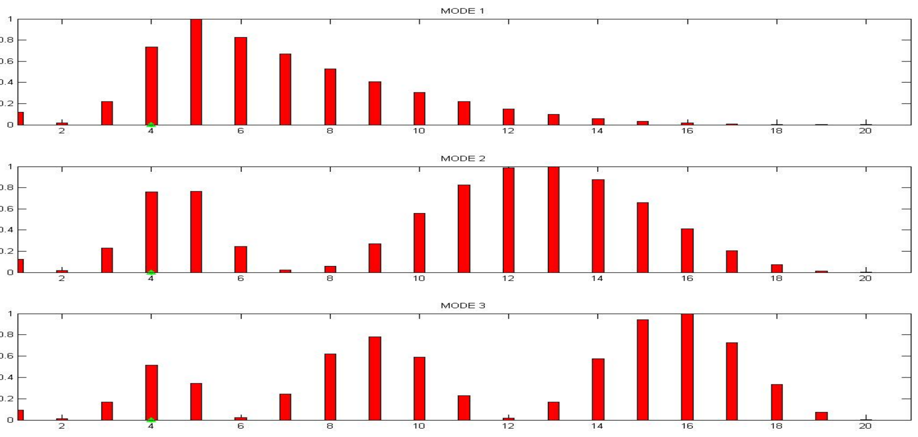


Figure 23. ABC with DOF 9 pinned, Modes 1-3, Stiffness

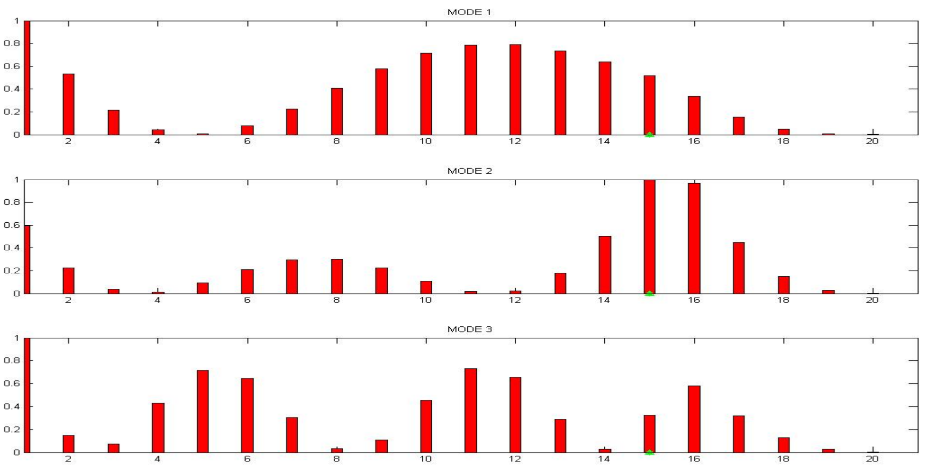


Figure 24. ABC with DOF 31 pinned, Modes 1-3, Stiffness

Although the exact damaged element was not found the general region was and the magnitude was as well. The system with ABC at Node 31 found the damage due to the high sensitivity in the region around element 15 with this particular boundary condition.

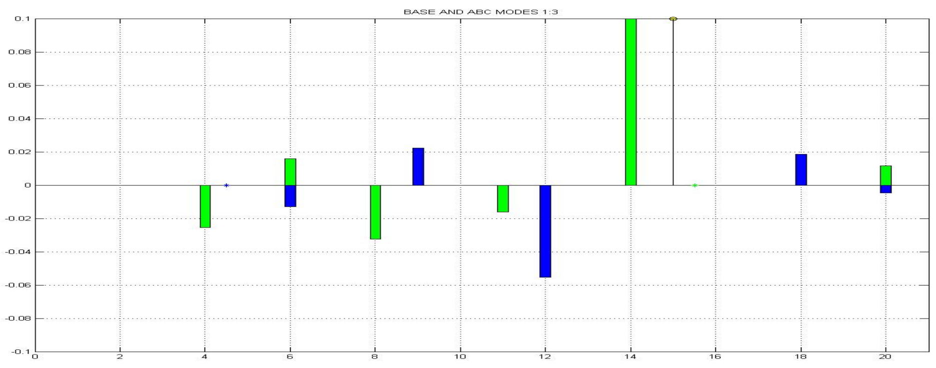


Figure 25. Element 15, Base and ABC systems, Modes 1:3, Mass

The solution for a change of mass demonstrates this same principal. The ABC applied further from the clamped end of the beam helped to even out the sensitivities of the beam and thus had improved error localization. The system with DOF 9 pinned resulted in similar sensitivities to that of the base system which did not help to strengthen the areas of weak sensitivity. Figures (26) through (28) show the base system, ABC with

DOF 9 pinned and ABC with DOF 31 pinned. Of particular interest is Figure (28) and how pinning DOF 31 lowered the sensitivity around element 15 and thus worsened the error prediction.

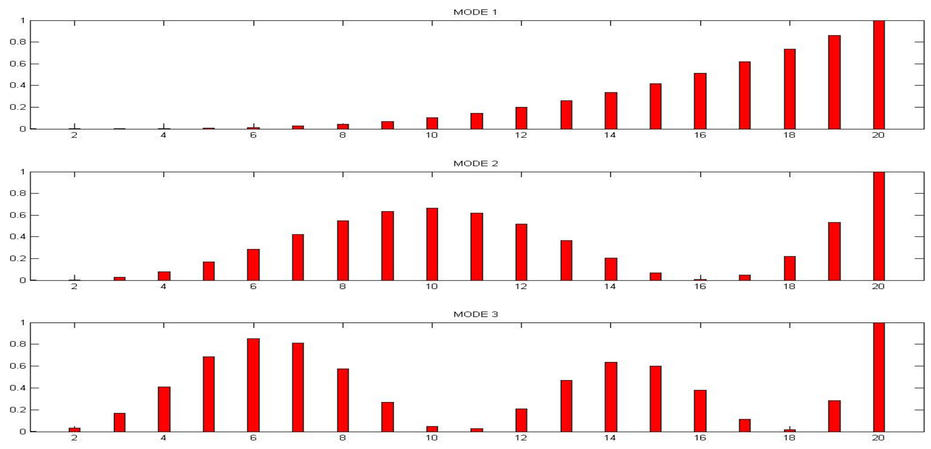


Figure 26. Base System, Modes 1-3, Mass

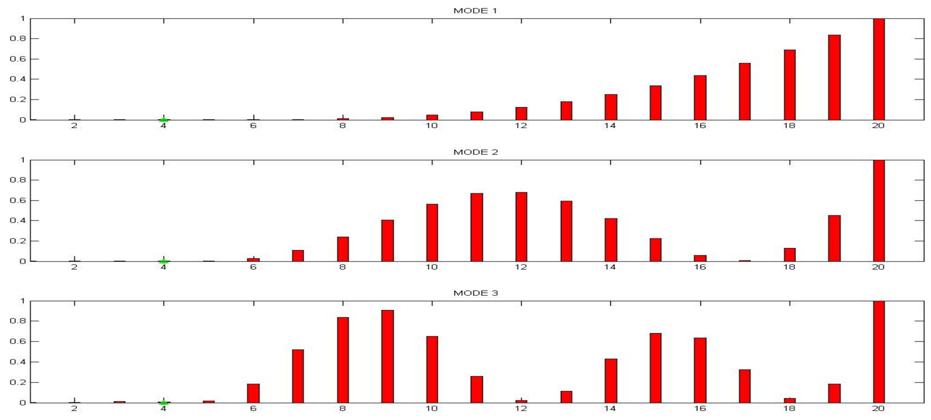


Figure 27. ABC with DOF 9 pinned, Modes 1-3, Mass

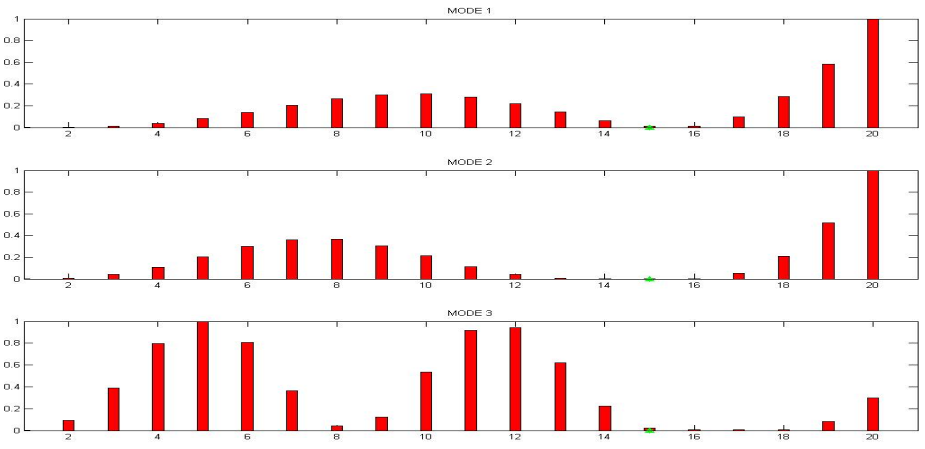


Figure 28. ABC with DOF 31 pinned, Modes 1-3, Mass

For all of the scenarios a pattern did emerge that depended on the sensitivity matrix. The magnitude of the perturbation did not have an effect on error location or detection. Rather the detection was a function of how sensitive the element was to an error. An element that was located in a region of high sensitivity resulted in relatively efficient error prediction. An element that was located in a region of low sensitivity had a difficult time in error prediction.

An additional study was done on the stiffness perturbation. A general solution program procedure was proposed. By running the program with all ABC applied to the system one at a time it was found that the error could be located by looking for the largest change in design variable across the results. Figure (29) below shows the beam with all the possible ABC pins. For example a change in stiffness was applied to element 10, the program was then run for all systems, base and ABC. Using just mode number one it can be shown that use of additional boundary conditions can isolate the error. Figure (29 below shows where each of the boundary conditions were applied one at a time.

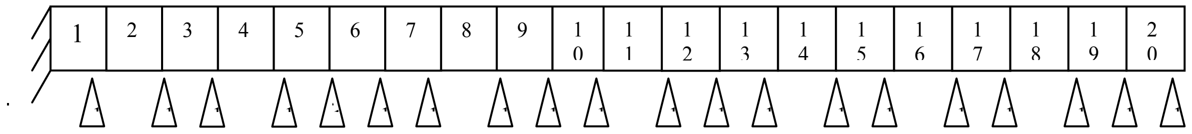


Figure 29. Application of pinned ABC starting with clamped end of beam.

Element	Base	ABC 3	ABC 5	ABC 7	ABC 9	ABC 11	ABC 13	ABC 15	ABC 17	ABC 19
1	0.0133	0	0	0	0	0	0	0	0	0
2	0	0.0158	0	0	0	0	0	0	0	0
3	0	0	0.0189	0	0	0	0	0	0	0
4	0	0	0	0.0228	0	0	0	0	0	0
5	0	0	0	0	0.0278	0	0	0	0	0
6	0	0	0	0	0	0.0344	0	0	0	0
7	0	0	0	0	0	0	0.043	0	0	0
8	0	0	0	0	0	0	0	0.0545	0	0
9	0	0	0	0	0	0	0	0	0.0703	0
10	0	0	0	0	0	0	0	0	0	0.0924
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0

Element	ABC 21	ABC 23	ABC 25	ABC 27	ABC 29	ABC 31	ABC 33	ABC 35	ABC 37	ABC 39	ABC 41
1	0	0	0	0	0	0.0654	0.0543	0.0486	0.0437	0.0387	0.0336
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
11	0.0923	0.0689	0	0	0	0	0	0	0	0	0
12	0	0	0.0547	0	0	0	0	0	0	0	0

13	0	0	0	0.0496	0	0	0	0	0	0	0
14	0	0	0	0	0.0602	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0

Table 5. Error prediction using one mode of each ABC system.

As the ABC approaches the error the value of the change in design variable increases, then as the ABC passes the value decreases. With this method the error was located and the value predicted to within 7 percent with the use of only one mode. The behavior of the system at the end of the beam created the poor prediction of the error after ABC 29, but did not affect the solution for errors located in other regions. In the case where the error was located in this region this solution method would not be an affective tool. A solution method of this type was not found to be applicable to mass perturbations.

C. OPTIMIZATION PROGRAM

Due to the relative difficulty of predicting mass error in the structure an optimization program was developed that utilized both frequency change and mode shape comparison to help locate the error. The mode shape data was assessed using the modal assurance criterion (MAC)(Allemang, 1982).

$$MAC = \frac{\left|\sum_{l=1}^{p}\left\{\Phi_{l,i}^{e}\right\}^{T}\left\{\Phi_{l,i}^{e}\right\}\right|^{2}}{\left(\sum_{l=1}^{p}\left\{\Phi_{l,i}^{e}\right\}^{T}\left\{\Phi_{l,i}^{e}\right\}\right)\left(\sum_{l=1}^{p}\left\{\Phi_{l,i}^{e}\right\}^{T}\left\{\Phi_{l,i}^{e}\right\}\right)} \qquad (57)$$

where $\left\{\Phi_{l,i}^{e}\right\}$ is the modal amplitude at location l of the *i*th experimental mode; $\left\{\Phi_{l,j}^{a}\right\}$ the modal amplitude at location l of the *j*th calculated mode; and p the length of the modal shape vector (Zhang, 2000). The experimental eigenvalues and eigenvectors were extracted from the FRFs taken from the experimental set up described in Appendix A which matches the FEM created for the previous error prediction calculations.

The frequency difference and mode shape data were combined in the objective function below.

$$J = \sum_{l=1}^{p}\left|\frac{\lambda_l^e - \lambda_l^a}{\lambda_l^e}\right| + \left(1 - \frac{\left|\sum_{l=1}^{p}\left\{\Phi_{l,i}^e\right\}^T\left\{\Phi_{l,i}^e\right\}\right|^2}{\left(\sum_{l=1}^{p}\left\{\Phi_{l,i}^e\right\}^T\left\{\Phi_{l,i}^e\right\}\right)\left(\sum_{l=1}^{p}\left\{\Phi_{l,i}^e\right\}^T\left\{\Phi_{l,i}^e\right\}\right)}\right) \tag{58}$$

The MAC produces values from 0 to 1, 1 indicating the two mode shapes being compared are quite close. Thus, the MAC was subtracted from one to produce a result that would indicate a better performance when minimized. The function was minimized by use of the MATLAB algorithm "fmincom". This program allows the use of lower and upper bounds, inequalities and equalities. Of interest was the use of just the mode shapes or frequencies of the system and if the combination of the two would produce a more reliable and accurate solution. In addition a variety of constraints were used in performing the computations to try and improve the performance of the objective function.

The problem was run with no constraints and then constrained. The initial constraints used in the problem were the amount of mass that could be added to any one element as well as the total amount of mass that could be added to the system. Further constraints that were recognized were the limitation of mass addition to only certain elements.

1. Procedure

To validate the optimization program initial trials were run with comparing two MATLAB beams, one with an error, the other with no error. The error was applied to element 35 of the model. The magnitude of the error was 10 percent of the total beam weight. Three separate objective functions were used in the comparison. The first was just the comparison of natural frequencies,

$$min \quad J = \sum_{l=1}^{p} \left| \frac{\lambda_i^e - \lambda_i^a}{\lambda_i^e} \right| \tag{60}$$

the second a comparison of mode shapes (MAC),

$$min \quad J = \sum_{l=1}^{p} \left(1 - \frac{\left| \sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,j}^a \right\} \right|^2}{\left(\sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right) \left(\sum_{l=1}^{p} \left\{ \Phi_{l,j}^a \right\}^T \left\{ \Phi_{l,j}^a \right\} \right)} \right) \tag{61}$$

and the third comparison of both natural frequencies and mode shapes (MAC).

$$min \quad J = \sum_{l=1}^{p} \left| \frac{\lambda_l^e - \lambda_l^a}{\lambda_l^e} \right| + \left(1 - \frac{\left| \sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right|^2}{\left(\sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right) \left(\sum_{l=1}^{p} \left\{ \Phi_{l,i}^e \right\}^T \left\{ \Phi_{l,i}^e \right\} \right)} \right) \tag{62}$$

The above objective functions were minimized subject to the following constraints.

1. Unconstrained

or

2. Limed amount of mass that could be added to any one element (10 percent).

or

3. Limited total amount of mass that could be added to the entire beam (10 percent).

Once the MATLAB comparison was done the same program was utilized in comparing experimental data from the laboratory. The procedure for extracting the modal parameters from the beam follows.

The FRFs were taken from the laboratory beam at DOF 42. This data was considered the base line data. Each DOF was one inch apart and DOF 1 was at the clamped end of the beam. The natural frequencies and mode shapes extracted using the RT Pro Focus and ME'Scope'VES curve fit the data. The curve fitting procedure uses a mathematical algorithm to estimate the modal parameters from the measured data. The mode shapes extracted from the laboratory software consisted of real and imaginary data. An algorithm in the software converted this data to real mode shapes that can be compared to the FEM data produced in MATLAB. The algorithm takes the sum of the squares of the real and imaginary portions for the magnitude, then assigns a phase angle of either 0 or 180 degrees. This is also known as the "Simple Method" (Ewins, 2005)

There was an initial difference in natural frequencies between the laboratory beam and the FEM that varied between approximately three and five percent. Due to this difference the optimization program was run to update the FEM to match the natural frequencies of the actual beam set up.

Next a mass error was added to the laboratory beam at a particular element. The FRFs were taken for all 42 DOF and natural frequencies and mode shapes extracted. The optimization program was then run again with the updated FEM to locate the mass error.

2. Results

Full results are available in Appendix B.

1. Natural Frequency only: The natural frequency used alone did not provide highly accurate error predictions. The best error prediction it could accomplish was within 74 percent of the actual error. For several of the cases the number of iterations were exceeded so the program stopped prior to meeting the desired tolerance.
2. Mode shapes only: Overall, the best performance was from the use of the MAC (mode shapes alone). This was the case for both experimental and MATLAB data. Additionally, as the problem became more constrained the error prediction became better. The best performance for this was the fully constrained problem with constraints on both total mass added to the system and total mass allowed for each element. This gave error predictions within 5 percent for the MATLAB portion and 9 percent for the experimental.

3. Mode shapes and natural frequencies: The use of both modes shapes and natural frequencies in the objective function did improve the results from just the use of natural frequencies, but did not improve the results from just using mode shapes alone.

The optimization program did generate reliable data for the comparison of eigenvectors (mode shapes) however, did not present itself as a good error predictor when using only eigenvalues (natural frequencies). This was more than likely due to the fact that when comparing eigenvectors there is more information about the system being compared (displacement at each DOF). Where as when comparing eigenvalues the only comparison is of several values, each of which is described by the entire system. The use of only natural frequencies in damage detection has been known to have limitations (Salawu, 1997).

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND RECOMMENDATIONS

The use of Artificial Boundary Conditions in sensitivity-based model updating has proven to be an effective method of overcoming the shortfalls of an underdetermined problem. The focus of this thesis was to try and evaluate and obtain reliable methods to help generate an improved error location method.

A. CONCLUSIONS

For the system tested in this thesis, with respect to perturbations or errors in mass and stiffness, the performance of the updating could be greatly improved by judicious application of ABC. For both mass and stiffness errors the best results were obtained when the sensitivities of the system were more evenly distributed across the beam. When a system had areas of low sensitivities it created the possibility of poor error detection. This could be reduced by increasing the sensitivity in these regions. This could be accomplished by applying an ABC in a position that would increase the sensitivity in the region(s) of poor sensitivity.

This was relatively easy to anticipate with the lower modes of the system. As the modes increased so too did the complexity of the mode shapes and sensitivity distribution. The sensitivity distribution with respect to stiffness was particularly reliable to anticipated, essentially knowing that the sensitivity would be increased in the region of the boundary conditions (pin). The sensitivity distributions with respect to mass were more difficult to predict. They tended to increase as the distance from a boundary condition increased which correspond to areas of large displacement as these are also areas of large acceleration and hence kinetic energy.

A better performing MATLAB computation method was not discovered. The QR Decomposition method produced results that often dropped out the desired solution. A correction for this discrepancy was not discovered.

The use of an optimization program to develop a solution method to a mass perturbation was also incorporated. This program made us of both natural frequencies and mode shape information. This was done in order to improve the performance of the error prediction.

B. RECOMMMENDATIONS

1. Investigate the properties of the QR decomposition algorithm which do not smear the solution. This could lead to an improved method of error location and retention with addition modes summed.

2. Comparison of mass and stiffness sensitivity matrices with strain and kinetic energy.

3. Develop an optimization program that would minimize the differences in modal parameters between a FEM and an actual structure with a stiffness error applied to the actual structure.

4. Further investigate the use of an optimization program to select ABC that would minimize areas of low sensitivity in a FEM to correlate to accurate model updating.

5. Test the use of sensitivity-based updating on a more complex base system (additional base boundary conditions) to evaluate the minimizing of regions of low sensitivity.

APPENDIX A

A. EXPERIMENTAL SET UP

A block of steel 18 inches in length, 8 inches wide and 2 inches thick was placed on a platform as a foundation. A cantilever beam made of T-6061 Aluminum, 48 inches in length, 1.5 inches wide and 0.5 inches thick, density of 0.11 lbf/in^3 and elasticity modulus of 10 E6 lbf/sec^2-in was placed on top of foundation steel table. The beam extended 42 inches and was clamped to the foundation with two large C clamps and two shorter length steel beams, each 6 inches long, 2 inches wide and 1 inch thick. These assisted in the elimination of platform generated modes.

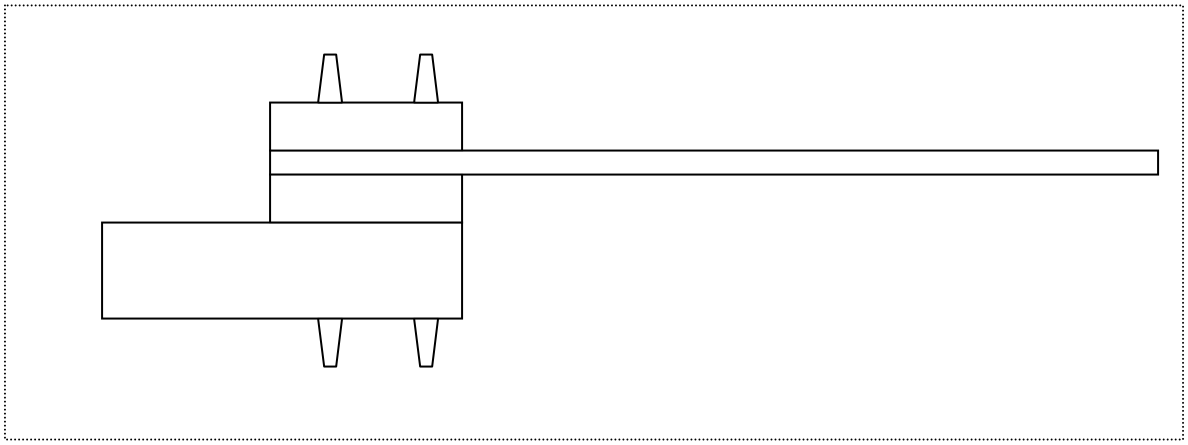


Figure 30. Experimental beam set up

The beam consisted of 42 elements, each 1 inch in length; this corresponded to FE model element quantity and length. A Series 336 FLEXCEL ICP accelerometer (serial number 10860) was threaded into position at node 41 of the beam and wired into Channel 2 on a DACTRON Focus front end digital signal processor (DSP). An excitation was applied by a PCB Series 086 B03 impact hammer (serial number 269), which was wired into Channel 1 on the DSP. The accelerometer and force hammer was calibrated using a pendulous test and the sensitivity adjustment was applied in the set-up of RT Pro Focus 5.57 software.

B. DATA COLLECTION

Using DACTRON RT Pro Focus 5.57 software, FRFs were collected when the roving force was applied by the load cell at each node. Node 41 remained the reference as the load cell roved from one node to next allowing for the measurement of the response at each node. Due to this set-up only one column or row of the complete FRF matrix [H] was actually measured.

1. A RT Pro Focus 5.57 software "Real-time" project was configured to measure 3200 spectral lines, 8192 points, with a delta T of 166.7μs over the frequency range 0-2400Hz. The frequency range of 0-2400 Hz was chosen because it covered the first 10 modes of system and signal resolution was sufficient for data acquisition. The excitation signal proved to be clean and thus no window was used for data measuring.

2. Channel set-up

	Channel 1 (Excitation)	Channel 2(Response)
Max Volts (mV):	0.1	0.3
Quantity:	Force	Accel.
EU:	lbf	gn
mv/EU:	8.67	104.383
Coupling:	ICP AC 0.7 Hz	ICP AC 0.7 Hz
Sensitivity Adjustment:	0	0

3. Trigger set-up

Source: Analog input

Run Mode: Manual Arm every frame

Input: Channel 1

Slope: Bi-polar

Level (%): 1, Level (V): 0

Pre/Post Points (-/+): -10

Pre/Post Time (-/+): -1.67µs

4. Average set-up:

Type: Linear

Domain: Frequency

Frames: 3 (Each node was excited 3 times and an average taken and saved.)

Accept/Reject: Manual Accept/Reject every frame.

(The user rejects double taps, under powered or overloaded signals.)

5. Modal Coordinate set-up:

Auto increment: ON

Rove: Excitation

Point increment: 1

Export: UFF text format, frequency response.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

Natural Frequencies						
	MATLAB			Experimental		
	Unconstrained	Constrained Element Mass	Constrained Total Mass	Unconstrained	Constrained Element Mass	Constrained Total Mass
Iterations	67	78	24	66	67	66
Fct. Ct	4242	4217	1321	4224	3490	4242
Element						
1	0.2098	0.0991	0	-0.0984	-0.1147	-0.1147
2	0	0.0732	0.0009	0.67	0.6608	0.8426
3	0	0	0	-0.1886	-0.1886	-0.1886
4	0	0	0	0.1173	0.1785	0.1173
5	0	0	0	0.0624	0.0624	0.0679
6	0	0	0.0374	0.0066	0.0066	0.0975
7	0.278	0.0695	0.0882	-0.0286	-0.0286	-0.0286
8	0.0932	0.2259	0	0.0933	0.0724	0.1614
9	0.004	0.0171	0	-0.0405	-0.047	-0.0156
10	0.15	0.0427	0	0.1819	0.1819	0.1819
11	0.0169	0.0447	0	0.1513	0.1513	0.1513
12	0	0	0	0.1409	0.1409	0.1409
13	0	0	0.035	-0.0506	-0.0506	-0.0506
14	0	0	0	0.1444	0.1444	0.1444
15	0	0	0	-0.0208	-0.0208	-0.0208
16	0.0013	0	0	0.184	0.0802	0.0802
17	0.2109	0.035	0	0.1242	0.1242	0.1242
18	0	0	0	0.0274	0.0274	0.0274
19	0	0	0	0.0584	0.0584	0.0584
20	0.013	0	0	-0.0352	-0.0352	-0.0352
21	0.0252	0	0.0153	-0.0103	-0.0103	-0.0103
22	0	0	0	0.1608	0.157	0.1445
23	0.0024	0	0	0.0431	0.0431	0.0431
24	0.1112	0	0.048	0.0821	0.0821	0.0907
25	0	0.0897	0	0.0663	0.0663	0.0663
26	0	0	0.039	-0.0768	-0.0768	-0.0768
27	0.0003	0.0432	0	0.6871	0.6015	0.6939
28	0	0	0.0352	0.1137	0.0581	0.0581
29	0.0479	0.0018	0	0.2309	0.2309	0.2309
30	0.1194	0.2332	0	0.2467	0.1548	0.1034
31	0.0007	0.0387	0	-0.1399	-0.1399	-0.1399
32	0.0604	0.1613	0	-0.0449	-0.0449	0.0274
33	0.1188	0	0	0.2998	0.3038	0.2633
34	0.7715	0.9104	0.3767	0.1642	0.144	0.2326
35	1.4782	1.539	0.2024	0.2385	0.3174	0.0821
36	0.0604	0.0793	0.3749	0.6164	0.7565	0.6955
37	0	0.0212	0.213	0.0312	0.0312	0.0312
38	0.0663	0	0.1581	0.0444	0.0444	0.0444
39	0.6677	0.7061	0	-0.0786	-0.0011	-0.0782
40	0.0459	0.0024	0	0.0186	0.1087	0.0186
41	0.0008	0.1152	0.4847	0.084	0.084	0.084
42	0	0	0.2686	0.2984	0.1897	0.2395

Table 6. Objective Function with Natural Frequencies

Mode Shapes						
	MATLAB			Experimental		
	Unconstrained	Constrained Element Mass	Constrained Total Mass	Unconstrained	Constrained Element Mass	Constrained Total Mass
Element						
1	0.0046	0.0046	0.0002	0.1565	0.3236	0
2	0.0416	0.0416	0	0.571	0.5263	0
3	0.1828	0.1828	1E-04	1.2007	1.0048	0
4	0.3305	0.3305	0.0778	1.6021	1.1174	0
5	0.3206	0.3206	0.0028	0.3853	0.3414	0.2939
6	0.1107	0.1107	0.0162	0.4479	0.0458	0
7	0.2706	0.2706	0.0144	1.0487	0.0882	0
8	0.3211	0.3211	0.0237	1.8214	0.7641	0
9	0.2001	0.2001	0.0164	1.6041	1.0079	0.2646
10	0.2068	0.2068	0.0685	1.1049	0.867	0
11	0.315	0.315	0	0.1776	0.0057	0
12	0.2615	0.2615	0.0105	1.3055	0.3166	0
13	0.2157	0.2157	0.0123	0.2975	0.0917	0
14	0.2587	0.2587	0.0028	0.2282	0.0032	0
15	0.2879	0.2879	0.0269	0.83	0.4894	0
16	0.2654	0.2654	0.0752	1.0372	0.4838	0
17	0.2133	0.2133	0	0.1547	0.0229	0
18	0.1869	0.1869	0	0.4684	0.035	0
19	0.2529	0.2529	0.0039	0.3138	0.0193	0
20	0.3319	0.3319	0.0848	0.9953	0.2707	0
21	0.2443	0.2443	0	1.0559	0.5797	0
22	0.1733	0.1733	0.012	0.3945	0.3839	0
23	0.2353	0.2353	0.0449	0.317	0.0012	0
24	0.3023	0.3023	0.0348	0.6827	0.0313	0
25	0.3175	0.3175	0.0063	0.8176	0.2061	0
26	0.2266	0.2266	0.0032	0.5868	0.2703	0
27	0.0536	0.0536	0	0.1869	0.1067	0
28	0.2986	0.2986	0.0859	0.1289	0.1753	0
29	0.4143	0.4143	0.0123	0.8184	0.0357	0
30	0.2596	0.2596	0.0021	0.5693	0.0059	0
31	0.1672	0.1672	0.0022	0.2602	0	0
32	0	0	0.0004	0.0071	0	0
33	0.0986	0.0986	0.0007	0	0	0
34	2.0037	2.0037	0.2836	4.8772	3.4007	0
35	2.8639	2.8639	3.9957	10.7002	4.2	3.819
36	1.8299	1.8299	0.1881	8.6623	4.2	1.1225
37	0.0407	0.0407	0.0043	0.4437	0.8355	0
38	0.0482	0.0482	0.019	0.0352	0	0
39	0.2049	0.2049	0.0098	0.0074	0	0
40	0.2846	0.2846	0.0166	0.2684	0	0
41	0.4631	0.4631	0.0132	1.7755	0.4011	0
42	0.1444	0.1444	0.0266	0.3576	0.3763	0

Table 7. Objective Function with Mode Shapes

Natural Frequencies and Mode Shapes						
	MATLAB			Experimental		
	Unconstrained	Constrained Element Mass	Constrained Total Mass	Unconstrained	Constrained Element Mass	Constrained Total Mass
Iterations	78	79	34	66	72	39
Fct. Ct	4242	4217	1806	2905	3247	1734
Element						
1	0.1498	0	0.2067	0.053	0.1121	0
2	0.3398	0.5238	0	0.0492	0.7151	0
3	0.0263	0.0218	0	1.0797	0.9828	0
4	0.1714	0.0049	0.2189	0.446	0.8307	0.1022
5	0	0	0.037	0.9709	0.4619	0.1764
6	0	0	0	0.6148	0.0831	0
7	0.0014	0.2368	0	0.4279	0.0745	0
8	0.0028	0	0	0.5951	0.6515	0
9	0	0.056	0	2.2734	1.0106	0.3224
10	0.2066	0	0	0.6441	0.8483	0.0031
11	0.0049	0	0	0.0183	0.0215	0
12	0	0	0	0.6911	0.2321	0
13	0	0	0	0.3131	0.0585	0
14	0.0099	0.0012	0	0.019	0.0412	0
15	0	0.0295	0	1.2532	0.521	0
16	0	0.0561	0	0.0997	0.3509	0
17	0.0414	0.0042	0	0.2084	0.0454	0
18	0.0008	0.0173	0	0.2054	0.0013	0
19	0.0198	0.4014	0	0.1378	0.0019	0
20	0.0003	0	0	1.2053	0.4178	0
21	0.0037	0.1091	0	0.162	0.3559	0
22	0	0	0	0.0983	0.4215	0
23	0.0125	0	0	0.8814	0.0105	0
24	0.0226	0.0586	0	0.39	0.0272	0
25	0.0342	0	0	0.0491	0.2321	0
26	0.0053	0.0048	0	0.3196	0.1301	0
27	0.1928	0	0	0.6125	0.1351	0
28	0.0214	0.0772	0	0.0172	0.1504	0
29	0	0.1595	0	0.237	0.0416	0
30	0.1097	0.0084	0	0.4457	0	0
31	0	0.0123	0	0.015	0	0
32	0.0082	0.0012	0.1026	0.0104	0	0
33	0.6786	0.7911	0	0.1034	0	0
34	0.0061	0	0.6801	3.2499	3.313	0
35	1.3859	1.1767	1.5291	9.295	4.2	3.5997
36	0.1608	0.07	0.3855	7.5685	4.2	1.2961
37	0.4339	0.5896	0	0.1267	0.6179	0
38	0.0053	0.0002	0	0.0594	0	0
39	1E-04	0	0	0.0199	0	0
40	0.9151	0	0	0.1146	0	0
41	0	0.1381	0	0.9672	0.2685	0
42	0	0	0.0725	0.4176	0.4014	0

Table 8. Objective Function with Natural Frequencies and Mode Shapes

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

A. ABCRUNTHRU_JRM

```
%  ********************  ABCrunTHRU_jrm.m  ***********************

% This program calculates the condition number of the following
% sensitivity matrices used to calculate the DV (error prediction).
% 1) Base system only 5 modes (underdetermined)
% 2) ABC system 10 modes
% 3) Base system 5 modes + 5 modes from ABC system
% The last system is calculated 3 times.  Once for modes 1-5,
% another for modes 6-10, and again for modes 11-15.
%
% This program is called from Build2Beams.m.
%
% Written by Constance R S Fernandez, Spring 2004
% Updated by John R Mentzer, Spring 2007

%INPUTS
% --------
% icnt_oset
% T_sens_tot
% vect_lam_tot

% OUTPUTS
% -------
% aa
% dv_a
% dv_b
% dv_c
% intervel
% mode
% startmode
% ten
% dv_cal_ABC - matrix
% dv_calABCten - matrix
% dv_cal_BasePlus - matrix
% cond_ABC - matrix
% cond_ABCten - matrix
% cond_basePlus - matrix

% ----INITIALIZATION-------

abc = 0;
ten = 1;
intervel = 1;
int_abc = 1;
for aa = 1:icnt_oset +1 % number of conditions (base + ABC)
    a_c = 1; % reinitialize for each ABC system

    for mode = 1:3 % 3 sets of modes per ABC system (10 element beam)
        % (modes 1:5, 6:10, 11:15)
        startmode = abc + a_c;

        dv_a = [startmode: startmode+4];    % modes
        dv_d1 = [ten:ten];     % mode 1
        dv_d2 = [ten+1:ten+1];   % mode 2
        dv_d3 = [ten+2:ten+2];   % mode 3
        dv_d4 = [ten+3:ten+3];   % mode 4
        dv_d5 = [ten+4:ten+4];   % mode 5


        dv_c = [ten:ten+9]; % the first 10 modes of each ABC system
        % (modes 1-10 only)

        if dv_a == [1:.25*size(T_sens_tot,2)]; % if sensitivity matrix
            %has only 5 rows than the modes used are all 5, else modes
            %used are first five (base) and a set of selected 5 modes
            %of ABC system for a total of 10 modes.
            dv_b = [dv_a];
        else
```

```
            dv_b = [1:.25*size(T_sens_tot,2), dv_a];
        end
        %---Base System only---(underdetermined)

        % save DV calculates of as matrix for plotting
        dv_cal_ABC(:,intervel) = T_sens_tot(dv_a,:)\vect_lam_tot(dv_a); hold on
        % condition number of Sensitivity matrix used in DV cal.
        cond_ABC(intervel,1) = cond(T_sens_tot(dv_a,:));
       % cond_ABC(intervel,1) =
cond((T_sens_tot(dv_a,:))*(T_sens_tot(dv_a,:))');%(Ttrans)(T)
%
%         %mode 1
%          dv_cal_ABC1 = T_sens_tot(dv_d1,:)\vect_lam_tot(dv_d1);
%         %mode 2
%          dv_cal_ABC2 = T_sens_tot(dv_d2,:)\vect_lam_tot(dv_d2);
%         %mode 3
%          dv_cal_ABC3 = T_sens_tot(dv_d3,:)\vect_lam_tot(dv_d3);
%         %mode 4
%          dv_cal_ABC4 = T_sens_tot(dv_d4,:)\vect_lam_tot(dv_d4);
%         %mode 5
%          dv_cal_ABC5 = T_sens_tot(dv_d5,:)\vect_lam_tot(dv_d5);


        % ---Base + 5 modes of ABC system ----
        dv_cal_BasePlus(:,intervel) = T_sens_tot(dv_b,:)\vect_lam_tot(dv_b);
       cond_basePlus(intervel, 1) = cond(T_sens_tot(dv_b,:));
       % cond_basePlus(intervel, 1) =
cond(((T_sens_tot(dv_b,:))*(T_sens_tot(dv_b,:))'));%(Ttrans)(T)

        intervel = intervel + 1;
        a_c = a_c + 5; % five modes used at a time

    end % "mode" loop

    % ---ABC system only ----



       dv_cal_ABCten(:,int_abc) = T_sens_tot(dv_c,:)\vect_lam_tot(dv_c);
       cond_ABCten(int_abc,1) = cond(T_sens_tot(dv_c,:));
        %cond_ABCten(int_abc,1) =
cond(((T_sens_tot(dv_c,:))*(T_sens_tot(dv_c,:))'));%(Ttrans)(T)

    int_abc = int_abc + 1;
    abc = abc + 39; % advances to next ABC system, must change number "19"
    % to reflect the number of DOF in beam.  This beam had 10 elements thus
    % 19 DOF.
    ten = ten + 39; % advances to next ABC system
end % "aa" loop


% ********************  END ABCrunTHRU_jrm.m  ***********************
```

B. ADDLUMPMASS_JRM

```
% ********************  AddLumpMass_jrm.m  **********************

%  This script constructs a vector of lumped masses
%  which is added to the diagonal of the BeamX mass matrix.
%          Mass added to [mx] in Assemble2Beams.m
%
%  Written by Prof J.H. Gordis

% Inputs needed
% ----------------
% num_elements
% mx

% Outputs
% ----------------
% mass_diag
% mass_node
% mass_coord
% mass_DOF
% mx - updated

disp(' ');disp(' ');
disp(' ******************************************************')
disp(' ****          Lumped mass addition to beams        ****')
disp(' ****  Lumpmass DOFs defined for UNRESTRAINED beams  ****')
disp(' ******************************************************')

% initalize
if exist('mass_diag') == 0;  % define and apply lumped mass vector.

add_mass = 'n';
add_mass = input(' Add lumped masses to BeamX ? (y/n) ','s');

% Initialize vector to add to [mx] diagonal.

mass_diag = zeros(2*(num_elements+1),1);

  while add_mass == 'y';

    mass_node = input(' Node number for lumped mass ? ');

    mass_coord = input(' Translation or Rotation for lumped mass ? (t/r) ','s');

    if mass_coord == 't'; % Translational lumped mass
        mass_DOF = 2 * mass_node - 1;
    elseif mass_coord == 'r'; % rotational lumped mass
     mass_DOF = 2 * mass_node;
 end

 mass_diag(mass_DOF) = input(' Enter value of mass/inertia (in "lbf-sec^2/in" ');
    % puts lumped mass on correct DOF
 add_mass = input(' Add another lumped mass ?  (y/n) ','s');
    % can continue adding mass until 'n' is entered

  end;   % End while loop

end;   % End exist('mass_diag')

mx = mx + diag(mass_diag);   % Add lumped masses to [mx]:

% *******************  END ADDLUMPMASS_jrm.M  **********************
```

C. ASSEMLESENS_JRM

```
% ********************  AssembleSens_jrm.m  ***********************
%
% This program assembles the total sensitivity matrix, T_sens_tot and
% total lam vector,  vect_lam_tot and assembles the relative frequency
% error between the natural frequencies of Beam A and Beam X
% Written by Constance R S Fernandez, Spring 2004
% Updated by John R Mentzer, Spring 2007



% INPUTS
% -------
% vect_lamx_oset
% vect_lam_oset
% vect_lam
% T_sens_oset
% T_sens
%
% OUTPUTS
% -------
% vect_OSET
% vect_lam_tot
% T_sens_tot
% freq_OSET
% freq_OSETx
% rel_freqERROR

vect_OSET = vect_lamx_oset - vect_lam_oset;
% lamx from actual beam with error oset, lam from FE model oset
% Creating a vector  of lam differences calculated (Lx-La)
if vect_OSET == 0;
    % when vector is empty at first, the total vector is equal to the
    % lam vector of Beam A, i.e., the first 19 values of vect_lam_tot are
    % the natural freq squared (rad^2/sec^2) of Beam A

    vect_lam_tot = vect_lam;
else
      vect_lam_tot = cat(1, vect_lam, vect_OSET);
end



if T_sens_oset == 0;

    T_sens_tot = T_sens;
else
    T_sens_tot = cat(1, T_sens, T_sens_oset);
end

freq_OSET = sqrt(abs(vect_OSET))/2/pi;
% Natural frequency vector of Beam A in Hz
freq_OSETx = sqrt(abs(vect_lamx_oset))/2/pi;
% Natural frequency vector of Beam X in Hz
rel_freqERROR = freq_OSET./freq_OSETx*100;
% Relative error between Beam A OSET natural freq and Beam X OSET natural Freq.

% ********************  END AssembleSens_jrm.m  **********************
```

D. BEAMPROPERTIES_JRM

```
% ********************  BeamProperties_jrm.m  ***********************

% This is the "props_file" to load nominal beam data.
% This program is called by BeamA_Prompt_crs to provide beam properties
% in order to build Beam A.

% This program was written by Constance Fernandez, Spring 2004
% This progam modified by John R Mentzer, Spring 2007
% Outputs
% -------
% depth
% width
% E
% rho
% total_length
% num_elements
% nominal_EI
% nominal_area
% nominal_density

% Following are actual measurements from experimental set-up cantilever
% beam.
 depth = .504;% in z-dir (inches)
 width = 1.506; % in y-dir (inches)
 E = 10e6;
    %E = 1.65e6; % lbf/sec^2-in (10e6-ksi)
    %(1bf/in^2 = 6894.76Pa)-> E(lbf/in^2) = ()Pa/6894.76
 rho =0.110460934; %0.098;% lbf/in^3

    % T6 temper alloys require a 35-ksi tensile strength, 30-ksi yield
    % strength and a 10e6-ksi elastic modulus. Alloy 6061-T6 has 1.0
    % pct magnesium, 0.6 pct silicon, 0.3 pct copper and 0.2 pct chromium.
    % It has a 45-ksi tensile strength and 35-ksi yield strength.1 The
    % machinability of aluminum alloys are high (300) compared to titanium
    % (40). Aluminum alloys can easily be bent and provide easy loading and
    % unloading of parts. Also, aluminum is a highly conductive metal
    % compared to titanium.

% all measurement of distance are in inches
 total_length    = 42;
 num_elements    = 20;
 nominal_EI      = (width * depth^3 / 12) * E;
 nominal_area    = depth * width;% in^2
 nominal_density = rho;% lbf/in^3

% ********************  END BeamProperties_jrm.m  **********************
```

E. BEAMX_PROMT_JRM

```
% *******************  BeamX_Prompt_jrm.m  **********************

% Written By Prof Gordis

% Inputs needed
% -----------------
% element_EI
% element_mass

% Outputs
% -----------------
% i_lbls
% change_mass, change_EI
% new_lbls
% updated element_mass in column 2
% updated element_EI in column 2
% mass_lbls - index for sensitivy matrix
% EI_lbls - index for sensitity matrix
% dv_EI
% dv_mass
% dv_tot

% ___________________________________________________________________________
%
%                    Prompt User for BeamX Modification Data
% ___________________________________________________________________________


disp(' ');disp(' ');
disp(' Modify nominal physical properties for second beam')
disp(' ~~~~~~ ~~~~~~~ ~~~~~~~~ ~~~~~~~~~~ ~~~ ~~~~~~ ~~~~')

% Adjust mass values for second beam:
i_lbls = 0;

dv_mass =[];
mass_lbls = [];
change_mass = 'n';
change_mass = input('  Modify single/range element mass values (y/n)? ','s');
% user input
while change_mass ~= 'n';

    disp('   Enter element label(s) for mass modification')
    disp('   Use MATLAB vector format> 1 3 5:7 9  ')
    new_lbls = input('   >> ','s');
    new_lbls = eval(['[',new_lbls,']']); % Converts string to vector of labels

    i_lbls = i_lbls + 1;

    % CRS addition

    mass_lbls(i_lbls,1:length(new_lbls))= new_lbls; % index for sensitivity matrix

    disp('  Enter mass change for element range')
    mass_change = input('  Enter percentage mass change (+/- %) ');

    dv_mass(i_lbls,1) = mass_change/100; % vector of mass changes to second beam
(BeamX)

    element_mass(new_lbls,2) = element_mass(new_lbls,2)+...
        (mass_change/100) * element_mass(new_lbls,2);

    disp(' ')
    change_mass = input('  Modify another element mass value (y/n)? ','s');
    disp(' ')

end; % end while

% Adjust EI values for second beam:
i_lbls = 0;

change_EI = 'n';
```

```
dv_EI =[];
EI_lbls = [];

disp(' ')
change_EI = input('  Modify single/range element EI values (y/n)? ','s');
while change_EI ~= 'n';

    disp('   Enter element label(s) for EI modification')
    disp('   Use MATLAB vector format> 1 3 5:7 9  ')
    new_lbls = input('   >> ','s');
    new_lbls = eval(['[',new_lbls,']']); % Converts string to vector of labels

    i_lbls = i_lbls + 1;

    EI_lbls(i_lbls,1:length(new_lbls)) = new_lbls; % index for sensitivity matrix

    disp('  Enter EI change for element range')
    EI_change = input('  Enter percentage EI change (+/- %) ');

    dv_EI(i_lbls,1) = EI_change/100; % vector of EI changes on second Beam

    element_EI(new_lbls,2) = element_EI(new_lbls,2)+...
        (EI_change/100) * element_EI(new_lbls,2);

    disp(' ')
    change_EI = input('  Modify another element EI value (y/n)? ','s');
    disp(' ')
end; % end while

dv_tot = [dv_mass;dv_EI];
% vector of total changes to second beam (BeamX) but not location.

% End BeamX_Prompt.m
clear EI_change mass_change

% ********************  END BeamX_Prompt_jrm.m  ***********************
```

F. BOUNDARYCONDITIONS_JRM

```
% *******************  BoundaryConditions_jrm.m  **********************

% Written by Prof Gordis

% This script prompts the user boundary condition information
% The script creates a vector of DOF (with respect to the unrestrained
% structure) and then extracts the rows and columns of the complementary
% DOF.

%  Script defines vector "free_dof_set" containing
%  list of unrestrained dof.

% The boundary conditions are applied in this script.

% Inputs needed:
% -------------
% ndof
% ka, ma, kx, mx

% Outputs:
% -------------
% free_dof_set
% updated ka, ma, kx, mx
% icnt_dof
% add_dof
% bc_node
% bc_coord
% bc_DOF
% bc_boolean
% all_dofs
% restraint_switch
% ***************************************************
% Start code:


if exist('free_dof_set')==0;    %  Build free_dof_set vector

    disp(' Select a boundary condition set:')
    disp('    (1) Clamped-free')
    disp('    (2) Clamped-Clamped')
    disp('    (3) Pinned-Pinned')
    disp('    (4) User-Defined')
    disp('    (5) Free-Free')

    BC_Choice = input(' >> Enter choice: ');

    if BC_Choice == 1;      % Clamped-free _____________________________
        free_dof_set = [3:ndof];
        restraint_switch = 'y';

    elseif BC_Choice == 2;  % Clamped-Clamped __________________________

        free_dof_set = [3:ndof-2];
        restraint_switch = 'y';

    elseif BC_Choice == 3;  % Pinned-Pinned ____________________________

        free_dof_set = [2:ndof-2  ndof];
        restraint_switch = 'y';

    elseif BC_Choice == 4;  % User-Defined _____________________________

        icnt_dof = 0;
        add_dof = 'y';
        while add_dof == 'y';

            bc_node = input(' Node number for restraint ? "0" to end: ');

            if bc_node == 0;
                break
            end;
            bc_coord = input(' Translation or Rotation ? (t/r) ','s');

            icnt_dof = icnt_dof + 1;
            if bc_coord == 't';
                bc_DOF(icnt_dof) = 2 * bc_node - 1;
            elseif bc_coord == 'r';
                bc_DOF(icnt_dof) = 2 * bc_node;
            end;   % End if-else block

        end;  % End while add_dof

        bc_boolean = ones(ndof,1);                   % [1 1 1 ... icnt_dof]
        bc_boolean(bc_DOF) = zeros(length(bc_DOF),1);% Put zeros in restrained dof
        all_dofs = [1:ndof];                         % List of all dof
        free_dof_set = all_dofs(logical(bc_boolean));% Extract free dof
        restraint_switch = 'y';

    elseif BC_Choice == 5; % Free-free beam _______________________________

        free_dof_set = [1:ndof];
        restraint_switch = 'n';

    end;                    % End if-elseif choice block __________________

end;      % End exist block

ka = ka(free_dof_set,free_dof_set);
ma = ma(free_dof_set,free_dof_set);
kx = kx(free_dof_set,free_dof_set);
mx = mx(free_dof_set,free_dof_set);

% ****************END BoundaryConditions_jrm.m  ***********************
```

G. BUILD2BEAMS_JRM

```
% ********************  Build2Beams_jrm.m  **********************

clear
clc
% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 9/22/94  Basic two beam assembly
%       2.0:          Added multi-element changes
%       2.1  3/28/95  Added read/write to file, rebuild capability
%       2.2  3/29/95  Added lumped mass additions
%            3/10/04  Added Sensitivity matrices, error prediction, plots
%       2.3  5/10/07  Added Error predictor

% ***********************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~
%
%  This program assembles the mass and stiffness matrices for 2 free-free
%  beams, referred to as "BeamA" (analysis) and "BeamX" (experimental). The
%  program can be run in several modes:
%
%  "Build" mode:
%   ~~~~~  ~~~~~
%  The user provides baseline data for BeamA, assumed to be a
%  homogeneous, uniform beam. Data provided:
%
%     (1) Beam length
%     (2) Number of elements
%     (3) Nominal EI
%     (4) Nominal cross-sectional area
%     (5) Nominal weight density
%
%  The program then prompts the user for instructions on how to modify
%  "BeamA" data to arrive at "BeamX" data. The user can modify element
%  masses, and/or element EI values. The modification can be applied to
%  either a single element, or range of elements, e.g.
%
%      Modify single/range element mass values (y/n)?  y
%
%  If "y" is entered, the user enters the number of the element for mass
%  adjustment:
%
%       Enter element label(s) for mass modification:  1
%       Use MATLAB vector format> 1 3 5:7 9
%
%               Enter percentage mass change (+/- %)
%
% The user is prompted to modify another element or range of elements:
%
%       Modify another element mass value (y/n)?  y
%
%  This process continues until the user enters an "n" for no change.
%  This entire process can then be repeated for EI adjustment.
%
%  The program saves the beam definition data in a binary (.mat) file
%  "beamdata" at the end of execution.
%
%  The program can also be run in "Read" mode by entering an "r" at
%  the initial prompt.
%
%
% Script Execution Path:
% ~~~~~~ ~~~~~~~~~ ~~~~~
%
%
%
%     Build2Beams_jrm.m         -- User executes this program.
%     BeamA_Prompt_jrm.m        -- Prompts User for BeamA nominal beam data
%     BeamX_Prompt_jrm.m        -- Prompts User for BeamX modification beam data
%     Assemble2Beams_jrm.m      -- Called by Build2Beams, builds [ka] [ma] [kx]
%                             [mx], plots freqs.
%     AddLumpmass_jrm.m             -- Prompts User for BeamX lumped mass addition
%     BoundaryConditions_jrm.m  -- Prompts user for B.C.'s and applies them.
%     PlotBeamModes_jrm.m       -- Calculate beam modes and plot frequencies
```

```
%
%     BeamSensitivity_jrm.m     -- Calculate sensitivity matrix T-sens
%     BeamSensitivityOSET_jrm.m -- Calculate sensitivity matrix using ABC
%     recorded_H_jrm.m          -- Calculates the nat. freq of BeamX with ABC
applied
%     AssembleSens_jrm.m        -- Assembles the sens matrices and calculates
errors.
%     ABCrunTHRU.m              -- Calculates the DV and cond number of matrix used
%     Saves data to "beamdata.mat"

%
%  Start code:
%  ~~~~~ ~~~~~
% *************************************************************************


disp('   Building 2 beams from scratch...')

BeamA_Prompt_jrm;       %      Prompt for BeamA Data: run prompt script
BeamX_Prompt_jrm;       %      Prompt for BeamX Modification Data:
Assemble2Beams_jrm;     %      Run script to assemble mass and stiffness matrices
AddLumpmass_jrm;        %      BeamX lumped mass vector construction and
%                           application

kx_beam = kx;  % saves the Beam X matrices without BC to be used later
mx_beam = mx;

BoundaryConditions_jrm; %      Prompt for, and apply boundary conditions

kx_beamBC = kx;  % saves the Beam X matrices with BC to be used later
mx_beamBC = mx;
ka_beamBC = ka;  % saves the Beam A matrices with BC to be used later
ma_beamBC = ma;

PlotBeamModes_crs       %  Calculate beam modes and plot frequencies

BeamSensitivity_jrm;    %  Calculate sensitivity matrix T-sens
BeamSensitivityOSET_jrm;%  Calculate sensitivity matrix using ABC


recorded_H_jrm;    % Calulates the nat. freq of BeamX with ABC applied
AssembleSens_jrm;  % Assembles the sens matrices and calculates errors.
ABCrunTHRU_jrm;    % Calculates the DV and cond number of matrix used
plottingBARS_jrm;      % Bar plots of predicted DV vs. true error
error_predictor;    % Generates error prediction for desired scenario.


%        Save Defining Parameters for Beams and plots
disp(' ...saving beam data to file')
save beamdata.mat


disp(' Build2Beams end.')
% ______________________

% ********************  END Build2Beams_jrm.m  **********************
```

H. DISPLACEMENTPLOT_OSET_JRM

```
% ********************  displacementPlot_OSET_jrm.m  ***********************

% Written by Constance Fernandez Spring 2004
% Updated by John Mentzer, Spring 2007

% This program plots the mode shapes (phi, lam) phi vs nodal position
% of beam when ABC are applied.

% Inputs
% -------
% plotkx
% kaO_base
% num_modesO
% phiXPLOT
% phiAPLOT
% pinned
% num_elements

% Outputs
% --------
%disp1, displa
%jj, g
% ypos

%------
% disp1 = zeros(ceil(.5*size(plotkx,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
disp1 = zeros(ceil((20/38)*size(plotkx,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
% disp1a = zeros(ceil(.5*size(kaO_base,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
disp1a = zeros(ceil((20/38)*size(kaO_base,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
for jj = 1:ceil((20/38)*size(plotkx,1));
    disp1(jj+1,:) = phiXPLOT(2*jj-1,1:num_modesO);
    % every other phi to give displacement at sequential nodes
    disp1a(jj+1,:) = phiAPLOT(2*jj-1,1:num_modesO);
end
% This loop normalizes the modes shapes to the tip modal displacement.
if pinned == 41 % tip pinned is a special case,  no new calculations are needed
   disp1(:,:) = disp1(:,:);
   disp1a(:,:) = disp1a(:,:);
elseif length(pinned)>1 & pinned(1,2)==41 %tip pinned with 2 ABC's
   disp1(:,:) = disp1(:,:);
   disp1a(:,:) = disp1a(:,:);
else
for g = 1:num_modesO
   disp1(:,g) = disp1(:,g)/disp1(num_elements+1,g);
   disp1a(:,g) = disp1a(:,g)/disp1a(num_elements+1,g);
end
end
% end
%ypos = [1:1:num_elements+1]; % Location of nodes used in plotting
ypos = [1:1:num_elements+1]; % Location of nodes used in plotting
% if mass_lbls ~= [];
%
%     for kk = 1:size(mass_lbls,1);
%         ff =0;
%         for JJ = 1:length(find(mass_lbls(kk,:)>0));
%             ff = ff+1;
%             posm(kk, 2*JJ-1) = mass_lbls(kk, ff);
%             posm(kk, 2*JJ) = mass_lbls(kk,ff)+1;
%         end
%     end
%
%     if kk == 1
%         posM = posm;
%
%     else
%
%         for uu = 1:kk-1;
%             posM = cat(2, posm(uu,:), posm(uu+1,:));
%         end
%
%     end
%     posM = sort(posM(find(posM>0)));
%     m = .5*ones(size(posM))+icnt_oset;
% end
%
% if EI_lbls ~= [];
%     for kk = 1:size(EI_lbls,1);
%         ff =0;
%         for JJ = 1:length(find(EI_lbls(kk,:)>0));
%             ff = ff+1;
%             pose(kk, 2*JJ-1) = EI_lbls(kk, ff);
%             pose(kk, 2*JJ) = EI_lbls(kk,ff)+1;
%         end
%     end
%
%     if kk == 1
%         posE= pose;
%
%     else
%
%         for uu = 1:kk-1;
%             posE = cat(2, pose(uu,:), pose(uu+1,:));
%         end
%     end
%   posE = sort(posE(find(posE>0)));
%   e = -.5*ones(size(posE))+icnt_oset;
% end


% ********************  END displacementPlot_OSET_jrm.m  ***********************
```

I. ERROR_PREDICTOR_JRM

```
%*************error_predictor_jrm.m****************

%Mode error predictor
%Written by John Mentzer, Spring 2007

% Dv=zeros(40,20);
% for n=1:40;
%     Dv(n,:)=(T_sens_tot(n,:)')*((lamx(n,:)/(2*pi))-(lama(n,:)/(2*pi)));
% end

format long
%//////////ORGINIAL METHOD////////////////////
for q=1:20
     dv_calculated_base2(:,q) = T_sens_tot(1:q,:)\vect_lam_tot(1:q);
end

for t=40:59
    dv_calculated_base3(:,(t-39)) = T_sens_tot(40:t,:)\vect_lam_tot(40:t);
end


%for BASE
for p=1:20
limit_cond(1,p)=cond(T_sens_tot(1:p,:));
end

%for ABC system

for ABCp=40:59
limit_condABC(1,(ABCp-39))=cond(T_sens_tot(40:ABCp,:));
end



% for g=1:20
%     dv_calculated_base_trans(g,:) = vect_lam_tot(g,:)*(T_sens_tot(g,:))';
% end
%
% figure(8)
% w=1:20
% bar(w,vect_lam_tot(w,1),.25,'r')
% % %
% dv_calculated_base11(:,1) = T_sens_tot(1,:)\vect_lam_tot(1);
% dv_calculated_base21(:,2) = T_sens_tot(2,:)\vect_lam_tot(2);
% dv_calculated_base31(:,3) = T_sens_tot(3,:)\vect_lam_tot(3);
% dv_calculated_base41(:,4) = T_sens_tot(4,:)\vect_lam_tot(4);
% dv_calculated_base51(:,5) = T_sens_tot(1:5,:)\vect_lam_tot(1:5);


dv_calculated_base2/(norm(dv_calculated_base2(:,:),inf));



%******************End of error_predictor_jrm.m*******************
```

J. FBEAMKM_JRM

```
% % ********************  fbeamkm_jrm.m  ***********************
%
% % function [kbeam,mbeam]=fbeamkm(l,ei,m)
% % Provided by Prof Gordis
%
% function [kbeam,mbeam]=fbeamkm(l,ei,m)
% %
% %
% % This function returns the stiffness and mass matrices for
% % a simple 2-node beam element.
% %
% % Note: m = rho * area * length = total element mass
% %
% % Reference: R.D. Cook, Concepts and Applications of F.E. Analysis
%
% % Outputs
% % ------
% % kbeam, mbeam, i, j
%
%
% kbeam=zeros(4,4);
% mbeam=zeros(4,4);
% %
% kbeam(1,1)=12.0;
% kbeam(1,2)=6.0*l;
% kbeam(1,3)=-12.0;
% kbeam(1,4)=6.0*l;
% kbeam(2,2)=4.0*l^2;
% kbeam(2,3)=-6.0*l;
% kbeam(2,4)=2.0*l^2;
% kbeam(3,3)=12.0;
% kbeam(3,4)=-6.0*l;
% kbeam(4,4)=4.0*l^2;
% %
% mbeam(1,1)=156.0;
% mbeam(1,2)=22.0*l;
% mbeam(1,3)=54.0;
% mbeam(1,4)=-13.0*l;
% mbeam(2,2)=4.0*l^2;
% mbeam(2,3)=13.0*l;
% mbeam(2,4)=-3.0*l^2;
% mbeam(3,3)=156.0;
% mbeam(3,4)=-22.0*l;
% mbeam(4,4)=4.0*l^2;
% %
% for i=1:4;
%     for j=i:4;
%         kbeam(j,i)=kbeam(i,j);
%         mbeam(j,i)=mbeam(i,j);
%     end
% end
% %
% kbeam=(ei/l^3)*kbeam;
% mbeam=(m/420.0)*mbeam;
% %
% % end function beamkm
%
% % ********************  END fbeamkm_jrm.m  ***********************
```

K. FMODES

```
% % ********************  fModes.m  ***********************
%
% function [lam,phi]=fmodes(k,m,num_to_print);
% % Provided by Prof Gordis
% % This program prints to the screen natural modes of system (phi).
% %
% %      Usage: [lam,phi]=fmodes(k,m,num_to_print)
% %
% %   This function can be used with 1 to 3 arguments, as follows:
% %
% %      [lam,phi]=fmodes(a)   : Produces modes of [a] with no print of freqs in
Hz.
% %      [lam,phi]=fmodes(a,i) : Produces modes of [a] with print of "i" freqs in
Hz.
% %      [lam,phi]=fmodes(k,m) : Produces modes of [m\k] with no print of freqs in
Hz.
% %
% %
% %
% %      This function returns a vector containing eigenvalues (rad/sec)^2
% %      and a matrix containing the mass normalized mode shapes.
% %      The mode information is sorted by frequency in ascending order.
% %      If num_to_print > 0; tabular listing of num_to_print freqs in Hz is
printed.
% %      If num_to_print <= 0, no print.
%
% % Inputs
% % ------
% % v, index, m, k
%
% % Programs
% % --------
% % fNormalize
%
% % Outputs
% % --------
% % phi
% % num_to_print
% % error
% % e
%
% %      --------------------------------------------------------------------------
-----
%
% if nargin == 1;
 %       [A] w/ no print request for freqs in Hz.
%
         %   v(1,:) = 1 normalization
%        [v,d]=eig(k);
%        [temp,indices] = sort(abs(diag(d)));
%        lam = diag(d);
%        lam = lam(indices);
%        [phi]=fNormalize(v(:,indices), 'one');
%        num_to_print = 0;
%
% elseif nargin == 2 & size(m,1) == 1;                      %       [A] w/ print
request for freqs in Hz.
%
         %   v(1,:) = 1 normalization
%        [v,d]=eig(k);
%        [temp,indices] = sort(abs(diag(d)));
%        lam = diag(d);
%        lam = lam(indices);
%        [phi]=fNormalize(v(:,indices), 'one');
%        num_to_print = m;
%
% elseif nargin == 2 & size(m,1) > 1;                               %       [k],[m]
w/ no print request for freqs in Hz.
%
         %   mass normalization
%        [v,d]=eig(m\k);
%        [lam,index]=sort(abs(diag(d)));
%        [phi]=fNormalize(v(:,index),'mass',m);
%        num_to_print = 0;
%
```

```
% elseif nargin == 3 & size(k,1) > 1 & size(m,1) > 1;%      [k],[m] w/ print
request for freqs in Hz.
%
        %   mass normalization
%       [v,d]=eig(m\k);
%       [lam,index]=sort(abs(diag(d)));
%       [phi]=fNormalize(v(:,index),'mass',m);
%
% else
%
%       num_to_print = -1;
%       error('Error from fModes.m: Check input arguments.')
%
% end
%
% if num_to_print > length(k);
%       num_to_print = length(k);
% end
%
% if nargin < 3 & rem(length(k),2)==0 & k(1:length(k)/2,1:length(k)/2) ==
zeros(length(k)/2,length(k)/2); % Have [A] matrix
%       e = 1;          % Eigenvalues are wn
% else
%       e = 0.5;    % Eigenvalues are wn^2
% end
%
%
%
% if num_to_print > 0;
%
%       disp('  '),disp('  ')
%       disp('~~~~~~~~~~~~~')
%       disp('Freqs in Hz.:')
%       disp((lam(1:num_to_print).^e)/2/pi)
%       disp('~~~~~~~~~~~~~')
%
% end
%
% % *******************  END fModes.m  **********************
```

L. FNORMALIZE

```
% % ********************  fNormalize.m  ***********************
%
% function [phi] = fNormalize(phi,method,m);
% %
% % Usage: [phi] = fNormalize(phi,method,m);
% %
% %      phi: matrix whose columns are to be (independently) normalized.
% %      method: String variable. The following choices are available:
% %
% %                    'mass'                Mass normalization
% %                    'inf'                 Infinity normalization
% %                    'one'                 First element = 1
% %                    'length'              Length = 1
% %
% %      m: matrix used for normalization, i.e., phi'*m*phi = eye
% %
% % _________________________________________
% %
%       switch method
%
%              case 'mass'                   % Mass normalization
%
% %                    disp('mass normalization')
%                      phi = phi * diag(sqrt(diag((phi' * m * phi).^(-1))));
%
%              case 'inf'                    % Infinity normalization
% %                    disp('inf normalization')
%                      for icnt_cols = 1:size(phi,2);
%                             phi(:,icnt_cols) =
phi(:,icnt_cols)/norm(phi(:,icnt_cols),inf);
%                      end
%
%              case 'one'                    % First element = 1
% %                    disp('one normalization')
%                      phi = phi * diag((phi(1,:).^(-1))');
%
%              case 'length'         % Length = 1
% %                    disp('length normalization')
%                      for icnt_cols = 1:size(phi,2);
%                             phi(:,icnt_cols) =
phi(:,icnt_cols)./norm(phi(:,icnt_cols),'fro');
%                      end
%
%       end
%
% % ********************  END fNormalize.m  ***********************
```

M. FOSET_FROM_ASET

```
% % ********************  fOset_from_Aset.m  ***********************
%
% function [oset] = fOset_from_Aset(ndof,aset);
% %
% %  Usage: [oset] = fOset_from_Aset(ndof,aset);
% %
% % This function determines the complementary subset "oset"
% % from a set [1:1:ndof] and the subset aset = [x x x ...].
% %
% %  ndof: Total number of DOF. Set is labeled "nset".
% %  aset: Retained DOF (proper subset of [1:1:ndof])
% %  oset: aset U oset = n
% %
% % Provided by Prof Gordis
% % ________________________________________________________
%
% nset = [1:ndof];
%
% for icnt = 1 : length(aset);
%      indices(icnt) = find(nset == aset(icnt));
% end
%
% bool = ones(size(nset));
% bool(indices) = zeros(size(indices));
% oset = nset(find(bool>0));
%
% % ********************  fOset_from_Aset.m  ***********************
```

N. FSPRINGMASS2

```
% % *******************  fSpringMass2.m  **********************
%
% function [k,m]=fSpringMass2(springs,mass,BC);
% %
% %      Usage: function [k,m]=fSpringMass2(springs,mass,BC)
% %
% %      This function script assembles the stiffness [k] and mass
% %      [m] matrices for an assemblage of springs.
% %
% %
% %      A linear chain of springs and masses is assumed.
% %             The number of springs is defined by the length of the vector
'springs'
% %             and their values by the elements of 'springs.'
% %
% %             The number of masses is defined by the length of the vector 'mass'
% %             and their values by the elements of 'mass'.
% %             NOTE:   The number of masses must equal to the final number of
active
% %                            DOF (i.e., after BC's applied).
% %
% %      Boundary conditions are specified by the vector 'BC.' This vector
% %      contains the DOF numbers which are to be restrained.
% %
% %      For example, to build the following system:
% %
% %                        .01        .02       .015
% %                |--////--[m]--////--[m]--////--[m]
% %                     5        6       3.4
% %
%       springs = [1 1 ];
%       mass    = [.01 .01 ];
%       BC             = [1]
% %
% %
% % _________________________________________________________
% %
% %                          BEGIN SCRIPT
% %                          ~~~~~ ~~~~~~
% %
%
% if length(mass) == (length(springs)+1) - length(BC);
%
%       k              = zeros(length(springs)+1,length(springs)+1);
%       m              = zeros(length(mass));
%
% %      assemble stiffness matrix:
%
%       rows = [0 1];
%       for ispring = 1 : length(springs);
%
%              rows = rows + 1;
%
%              addthis = [springs(ispring) -springs(ispring);-springs(ispring)
springs(ispring)];
%              k(rows,rows) = k(rows,rows) + addthis;
%       end
%
%       if ~isempty(BC);
%              keep = fOset_from_Aset(length(springs)+1,BC);
%              k = k(keep,keep);
%       end
%
% %      assemble mass matrix:
%
%       m = diag(mass);
%
% else
%
%       disp('Error in fSpringmass2. Check # masses, springs, and BC"s.')
%       return
%
% end
% % *******************  END fSpringMass2.m  **********************
%
```

O. NORMRUNTHRU_CRS

```
% ********************  normRUNthru_crs.m  ***********************

% This program finds the NORM of the columns of sensitivity matrix and the
% NORM of the rows of the inverse of the sensitivity matrix.  This was used
% to find a correlation of the good prediction to the ABC system used.  It
% also plots the information in helpful graphes.
%
% This program was written for a system of 19 natural freq. Set of 5 modes
% were used in each ABC system, i.e.,, modes 1-5, modes 6-10,or modes 11-15.
% This accounts for the 3 sets of modes per condition as listed below in
% the "for" loop modeN = 1:3. This program compares the use of the first 5
% modes of the base system and one set of five modes of the ABC  to that of
% of just 10 modes of the ABC.  Notice that the sensitivity is a 10x10
% square matrix indicating that only mass or EI changes, not both were
% made.

% This program is not part of Build2Beams_crs.m program.  It is run
% separately.

% Written by Constance R S Fernandez, Spring 2004

% Inputs
% ------
% cond_basePlus
% icnt_oset
% T_sens_tot
% EI_lbls, mass_lbls
% dv_mass, dv_EI, dv_cal_ABCten

% Outputs
% --------
% BASE
% BASET
% abcN, countN, a_cN
% modeN, startmodeN, startmodeNT
% bb, t, tINV, cc, T, TINV, vv, tt
% modelabelNORM
% norm_vectT, norm_vecTABC, norm_vecTinvABC
% normC
% baseN, baseABCN, abc_conN, abc_conTN
% baseABCNten, abc_conNten, abc_conTNten

% ----Start program----

BASE = int2str(cond_basePlus(1)); % cond no. of the base line system
BASET = sprintf('Base[1:5] Cond = %s', BASE); % used for plotting

% ======Initialization=====%

abcN = 0;
countN =0;

% =======Calculations of NORM vectors======%
for count = 1:icnt_oset +1 % number of conditions (base + ABC)
    a_cN = 1;

    for modeN = 1:3 % 3 sets of modes per boundry condition
        startmodeN = abcN + a_cN; % the beginning mode number of each set

        % indicates the use of the first 5 modes of base system + 5 modes
        % of ABC system
        bb = [1:5, startmodeN: startmodeN+4];

        % labeling of modes for plotting
        modelabelNORM = int2str(a_cN:a_cN+4);

        a_cN = a_cN+5;    %advances to the next set of modes

        % base system plus 5 modes of ABC
        t = T_sens_tot(bb,:); % 10x10 matrix
        tINV = inv(T_sens_tot(bb,:));% 10x10 matrix

        % first 10 modes of ABC solo
        startmodeNT = abcN+1; % the beginning mode number of each set
        cc = [startmodeNT: startmodeNT+9];
        T = T_sens_tot(cc,:);% 10x10 matrix
        TINV = inv(T_sens_tot(cc,:));% 10x10 matrix

        % for loop for NORM of columns and rows of inv(sens matrix)
        for vv = 1:10 % 10 rows, 10 columns
            % base + ABC system
            norm_vecT(vv,countN+modeN) = norm(t(:,vv)); % columns
            norm_vecTinv(vv,countN+modeN) = norm(tINV(vv,:)); % rows
            % for 10 modes ABC solo
            norm_vecTABC(vv,countN+modeN) = norm(T(:,vv)); % columns
            norm_vecTinvABC(vv,countN+modeN) = norm(TINV(vv,:)); % rows

        end % vv loop
    end % ModeN loop
    abcN = abcN+19; % advances to the next ABC system
    countN = countN+3; % counts up each set of ABC
end % count loop
normC=4; % initialize for plots

%==========PLOTTING==========%

for tt=1:10 % figures (30-40) plots 6 graphes per figure
    figure(tt+30)
    subplot(3,2,1)
    bar(norm_vecT(:,normC))
    title ('Norm col Tsens, ABC Modes [1:5]');

    subplot(3,2,3)
    bar(norm_vecTinv(:,normC))

    title ('Norm row TsensINV, ABC Modes [1:5]');

    subplot(3,2,2)
    bar(norm_vecTABC(:,normC))
    title ('Norm col Tsens, ABC Modes [1:10]')
    subplot(3,2,4)
    bar(norm_vecTinvABC(:,normC))
    title ('Norm row TsensINV, ABC Modes [1:10]')

    subplot(3,2,5)
    % plotting error prediction
    % using [1:5] modes of ABC system + [1:5] modes of Base system;
    baseABCN = bar(dv_cal_BasePlus(:,normC),.5,'r');hold on

    % plotting error prediction using [1:5] modes of Base system;
    baseN = bar(dv_cal_ABC(:,1),.25,'b');


    abc_conN = int2str(cond_basePlus(normC)); % cond no. for legend
    abc_conTN = sprintf('Base[1:5]+ABC[1:5] Cond = %s', abc_conN);

    grid on
    legend([baseABCN,baseN],BASET,abc_conTN), hold on


    % plotting actual error
    if EI_lbls ~=[] & mass_lbls ~=[]
        stem(mass_lbls, dv_mass,'y','filled'); hold on;
        stem(mass_lbls, dv_mass,'k')

        EIplot = EI_lbls+10;hold on
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    elseif  mass_lbls ~=[] & EI_lbls ==[]
        stem(mass_lbls, dv_mass,'y','filled');hold on;
        stem(mass_lbls, dv_mass,'k')

    else
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    end % if EI_lbls ~=[] & mass_lbls ~=[]

    title (sprintf('Error, Base [1:5] + ABC [1:5], pinned NODE # %d', (tt+1)))

    subplot(3,2,6)
    % plotting error prediction using [1:10] modes of ABC system;
    baseABCNten = bar(dv_cal_ABCten(:,normC));hold on
    abc_conNten = int2str(cond_ABCten(normC));
    abc_conTNten = sprintf('ABC[1:10] Cond = %s', abc_conNten);

    grid on
    legend([baseABCNten],abc_conTNten), hold on

    % plotting actual error
    if EI_lbls ~=[] & mass_lbls ~=[]
        stem(mass_lbls, dv_mass,'y','filled'); hold on;
        stem(mass_lbls, dv_mass,'k')

        EIplot = EI_lbls+10;hold on
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    elseif  mass_lbls ~=[] & EI_lbls ==[]
        stem(mass_lbls, dv_mass,'y','filled');hold on;
        stem(mass_lbls, dv_mass,'k')

    else
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k')

    end % EI_lbls ~=[] & mass_lbls ~=[]


    title (sprintf('Error, ABC Modes [1:10], pinned NODE # %d', (tt+1)))

    normC = normC+3; % advances to the next ABC system
end % tt = 1:10 for plotting graphes


% ********************  END normRUNthru_crs.m  ***********************
```

P. PLOTBEAMMODES_CRS

```
% *******************  PlotBeamModes_crs.m  **********************

% Calculates natural frequencies

% Provided by Prof Gordis

% Inputs needed:
% -----------------
% ka, ma, mx, kx

% Programs needed:
% -----------------
% fModes

% Outputs:
% -----------------
% lama, phia, lamx, phix (without rigid body modes)
% num_rbm
% phia_plot, phix_plot


disp('  ');
disp(' Calculating modes for each beam...plot frequency comparison')


% Get modes of each beam:

[lama,phia]=fModes(ka,ma);
[lamx,phix]=fModes(kx,mx);

%used to plot the mode shapes org BC before ABC
phia_plot = phia;
phix_plot = phix;

% Set any rigid body mode freqs to zero:

  num_rbm = length(find(lama < 1));

  sprintf('Number of Rigid Body Modes Found: %2i', num_rbm)

    disp( ' Removing rigid body mode frequencies from vectors...')
    lama = lama(find(lama > 1));
    lamx = lamx(find(lamx > 1));

% *******************  END PlotBeamModes_crs.m  **********************
```

Q. PLOTTINGBARS_CRS

```
% ********************  plottingBARS_crs.m  **********************

% To be used with Build2Beams.m and
%
% This program plots 9 graphes per figure.  The first columns of 3 graphes
% are the mode shapes of the ABC system used in error prediction.  The next
% column of 3 graphes are the error prediction using only 5 modes of ABC
% system. The last column of 3 graphes are the error predictions using the
% first 5 modes of base system plus 5 modes of the ABC system.  The row
% represent modes 1-5, middle row: modes 6-10, last row : modes 11-15. Each
% of the error prediction graphes also have the base only prediction  and
% the actual error plotted for easy reference.
%
% Written by Constance R S Fernandez, Spring 2004

% Inputs
% --------
% cond_basePlus
% FOM_ABC5per, FOM_PLUSper
% icnt_oset
% modeshape
% rel_freqERROR
% ypos
% EI_lbls, mass_lbls
% dv_mass, dv_EI

% Outputs
% --------
% BASE, BASET
% FOMBASE, FOMABC, FOMPLUS
% intervelp
% ER, barp, shape, error, a_cp, modep, ap,
% modelabelp, FOMABClabelp, FOMPLUSlabelp
% abc_con, abc_conT
% ABC, base
% plus_con, plus_conT
% base, plus, EI_plot


BASE = int2str(cond_basePlus(1));
FOMBASE = int2str(FOM_ABC5per(1));
BASET = sprintf('Base Cond = %s, FOM = %s', BASE, FOMBASE);


intervelp = 3;
modeshape = 1;
ER = 1;
for barp = 1:icnt_oset

    figure(barp+10) % figures 11-20
    format bank
    %shape = [modeshape:modeshape+20]; %This number is equal to number of elements.
    shape = [modeshape:modeshape+20]; %This number is equal to number of elements.
    error = round(rel_freqERROR(ER:ER+15)*100)/100;
    a_cp = 1;
    for modep = 1:3 %3 sets of modes per boundry condition

        ap = [a_cp: a_cp+4]; %modes
%         REL_error1 = int2str(error(a_cp));
%         Errorlabelp = sprintf('Rel error = %s', REL_error1);
%         REL_error2 = int2str(error(a_cp+1));
%          Errorlabelp = sprintf('Rel error = %s', REL_error2);
%         REL_error3 = int2str(error(a_cp+2));
%          Errorlabelp = sprintf('Rel error = %s', REL_error3);
%         REL_error4 = int2str(error(a_cp+3));
%          Errorlabelp = sprintf('Rel error = %s', REL_error4);
%         REL_error5 = int2str(error(a_cp+4));
%          Errorlabelp = sprintf('Rel error = %s', REL_error5);

        modelabelp = int2str(a_cp:a_cp+4);
        FOMABC = int2str(FOM_ABC5per(intervelp+modep));
        FOMABClabelp = sprintf('System FOM = %s', FOMABC);

        FOMPLUS = int2str(FOM_PLUSper(intervelp+modep));
        FOMPLUSlabelp = sprintf('System FOM = %s', FOMPLUS);
        % ============================================================%
        % =====mode shape or beam X and beam A with ABC=================%
        % ============================================================%
        figure(barp+50)
        subplot(3,1,modep)
        plot(ypos, dispA_tot(shape,a_cp),'k-o', ypos, ...
            dispA_tot(shape,a_cp+1),'g-s', ypos, ...
            dispA_tot(shape,a_cp+2),'b-d', ypos, ...
            dispA_tot(shape,a_cp+3),'r-x', ypos, ...
            dispA_tot(shape,a_cp+4),'m-*', ypos, ...
            dispX_tot(shape,a_cp),  'r--o', ypos, ...
            dispX_tot(shape,a_cp+1),'b--s', ypos, ...
            dispX_tot(shape,a_cp+2),'m--d', ypos, ...
            dispX_tot(shape,a_cp+3),'c--x', ypos, ...
            dispX_tot(shape,a_cp+4),'k--*'), grid on...


        legend(sprintf('Rel Freq Error = %d', error(a_cp)),...
            sprintf('Rel Freq Error = %d', error(a_cp+1)),...
            sprintf('Rel Freq Error = %d', error(a_cp+2)),...
            sprintf('Rel Freq Error = %d', error(a_cp+3)),...
            sprintf('Rel Freq Error = %d', error(a_cp+4)));

%         legend(sprintf('Bm X, Md %d', a_cp'), sprintf('Bm X, Md %d', a_cp+1),...
%             sprintf('Bm X, Md %d', a_cp+2), sprintf('Bm X, Md %d',a_cp+3),...
%             sprintf('Bm X, Md %d', a_cp+4), sprintf('Base, Md %d', a_cp), ...
%             sprintf('Base, Md %d', a_cp+1), sprintf('Base, Md %d', a_cp+2), ...
%             sprintf('Base, Md %d', a_cp+3), sprintf('Base, Md %d', a_cp+4))


        title(sprintf('Modes [ %s]',modelabelp))
        axis tight
        % ============================================================%
        % ===== bar graphes of error solution using only ABC=============%
        % ============================================================%
        figure(barp+10) % figures 11-20
        subplot(3,2,2*modep-1)

        abc_con = int2str(cond_ABC(intervelp+modep));
        abc_conT = sprintf('ABC Cond = %s, FOM = %s', abc_con, FOMABC);

        ABC = bar(dv_cal_ABC(:,intervelp+modep),.5,'r'); hold on
        %ABC = bar(dv_cal_ABC(:,1:5),.5,'r'); hold on  %trying to isolate the first
five modes

        base = bar(dv_cal_ABC(:,1),.25,'b');hold off%on % base first 5 modes
        %FOMabc = bar(1,0); hold off
        grid on
        %legend([ABC,base,FOMabc],plus_conT,BASET,FOMABClabelp), hold on
       % grid on
        legend([ABC,base],abc_conT,BASET), hold on

        title(sprintf('ABC only, [ %s]', modelabelp));




        if EI_lbls ~=0 & mass_lbls ~=0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled'); hold on;
            stem(mass_lbls, dv_mass,'k'), hold on;

            EIplot = EI_lbls+10; hold on % last half of plot
            stem(EIplot, dv_EI,'c','filled');hold on;
            stem(EIplot, dv_EI,'k'); hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*',...
             %   barp+11,0,'g^',barp+11,0,'gh',barp+11,0,'g*'  )
             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*',...
               ((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-1)/2),0,'g*'
)


        elseif  mass_lbls ~=0 %&EI_lbls =0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled'); hold on
            stem(mass_lbls, dv_mass,'k'); hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')
            plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        else
            % plots actual error
            stem(EI_lbls, dv_EI,'c','filled');hold on;
            stem(EI_lbls, dv_EI,'k'); hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')
            plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        end % EI_lbls ...

        % ===========================================================%
        % =========bar graphes of error solution using ABC + base========%
        % ===========================================================%
        subplot(3,2,2*modep)


        plus_con = int2str(cond_basePlus(intervelp+modep));% for legend
        plus_conT = sprintf('Base+ABC Cond = %s FOM = %s', plus_con, FOMPLUS);% for
legend

        plus = bar(dv_cal_BasePlus(:,intervelp+modep),.5,'r'); hold on
        base = bar(dv_cal_ABC(:,1),.25,'b'); hold on % base first 5 modes
        %FOMplus = bar(1,0); hold off
        grid on
        legend([plus,base],plus_conT,BASET), hold on
%        legend([plus,base,FOMplus],plus_conT,BASET,FOMPLUSlabelp), hold on

        if EI_lbls ~=0 & mass_lbls ~=0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled'); hold on;
            stem(mass_lbls, dv_mass,'k'); hold on
            EIplot = EI_lbls+10; hold on % last half of plot
            stem(EIplot, dv_EI,'c','filled');hold on;
            stem(EIplot, dv_EI,'k');hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*',...
             %   barp+11,0,'g^',barp+11,0,'gh',barp+11,0,'g*'  )
           plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*',...
               ((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-1)/2),0,'g*'
)

         elseif  mass_lbls ~=0 %&EI_lbls =0
            % plots actual error
            stem(mass_lbls, dv_mass,'y','filled');hold on;
            stem(mass_lbls, dv_mass,'k');hold on
            % plots the green triangle which indicates pinned node
            %plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')
             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        else
            % plots actual error
            stem(EI_lbls, dv_EI,'c','filled');hold on;
            stem(EI_lbls, dv_EI,'k'), hold on
            % plots the green triangle which indicates pinned node
           % plot(barp+9,0,'g^',barp+9,0,'gh',barp+9,0,'g*')%changed barp+1
             plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*')

        end % if EI_lbls ...

        title(sprintf('Base [1:5] + ABC [ %s]', modelabelp));
        %       title(sprintf('Base + ABC, pinned at NODE# %d',barp +1))
        a_cp= a_cp +5;
    end  % modep loop

    modeshape = modeshape + 11; % advances
    intervelp = intervelp +3; % advances to the next ABC system
    ER = ER+19;
end % barp loop
```

```
%  figure(barp+11)
%plot(1:20,ABC1, bar)


%  %ABC Mode 1 Error
%         subplot(5,1,1)
%         ABC1 =dv_cal_ABC1;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC1,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*');
%         title('ABC Mode 1')
% %ABC Mode 2 Error
%         subplot(5,1,2)
%
%         ABC2 = dv_cal_ABC2;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC2,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*') ;
%         title('ABC Mode 2')
%
% %ABC Mode 3 Error
%         subplot(5,1,3)
%
%         ABC3 = dv_cal_ABC3;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC3,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*') ;
%         title('ABC Mode 3')
%
%  %ABC Mode 4 Error
%         subplot(5,1,4)
%         ABC4 =dv_cal_ABC4;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC4,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*');
%         title('ABC Mode 4')
%
% %ABC Mode 5 Error
%         subplot(5,1,5)
%         ABC5 =dv_cal_ABC5;
%         stem(EI_lbls, dv_EI,'k'); hold on
%         bar(1:20,ABC5,.25,'r');
%         plot(((BC(:,3)-1)/2),0,'g^',((BC(:,3)-1)/2),0,'gh',((BC(:,3)-
1)/2),0,'g*');
%         title('ABC Mode 5')






% ********************  END plottingBARS_crs.m  ***********************
```

LIST OF REFERENCES

Allemang, R.J., Brown, D.L. A correlation coefficient for modal vector analysis. *Proceedings of 1st International Modal Analysis Conference*, 1982, 110-116.

Anton, H., Rorres, C., (2005). Elementary Linear Algebra. New York: John Wiley and Sons.

Avitable, Peter (2001, January) Experimental Modal Analysis, A Simple Non-Mathematical Presentation. *Sound and Vibration*, 1-11.

Bock, R. K. Home page. 7 April 1998, updated 6 May 2005. http://rkb.home.cern.ch/rkb/AN16pp/node265.htm.

Brownjohn, J.M.W., Xia, Pin-Qi, Hao, Hong, & Xia, Yong. (2001) Civil structure condition assessment by FE model updating methodology and case studies. Finite *Elements in Analysis and Design 37* (211) 761-775.

Ewins, D. J. (1984). *Modal Testing: Theory and Practice*. England: Research Studies Press LTD.

Fernandez, C., (2005) Artificial Boundary Conditions in Sensitivity Based Finite Element Model Updating and Structural Damage Detection.

Gordis, J. H. (1993) Spatial Frequency Domain Updating of Linear, Structural Dynamic Models

Gordis, J. H. (1994, February 21). Structural Synthesis in the Frequency Domain: A General Formulation. *Shock and Vibration. 1*(5) 461-471

Gordis, J. H. (1996, July). Omitted Coordinate Systems and Artificial Constraints in Spatially Incomplete Identification. *Modal Analysis: the International Journal of Analytical and Experimental Modal Analysis. 11*(1), 83-95

Gordis, J. H. (1999) Artificial Boundary Conditions for Model Updating and Damage Detention. *Mechanical Systems and Signal Processing*, *13* (3), 437-448

Hanson, D., Waters, T.P., Thompson, D.J., Randall, R.B., & Ford, R.A.J., (2006). The Role of Anti-Resonance Frequencies from Operating Modal Analysis in Finite Element Model Updating. *Mechanical Systems and Signal Processing 21*, 74-97.

Jaishi, Bijaya, Ren, Wei-Xin, (2005) Damage Detection by Finite Element Model Updating Using Modal Flexibility Residual. Journal *of Sound and Vibration 290*, 369-387.

Mottershead, J.E., (1993) Model Updating in Structural Dynamics: A Survey. *Journal of Sound and Vibration 167* (2), 347-375.

Panday, A.K., & Biswas, M., (1991). Damage Detection in Structures Using Changes in Flexibility. *Journal of Sound and Vibration 169* (1), 3-17.

Watkins, D.S., (2002). *Fundamentals of Matrix Computations*. New York: John Wiley and Sons.

Zhang, Q.W., Chang, C.C., & Chang, T.Y.P., (2000). Finite Element Model Updating for Structures with Parametric Constraints. *Earthquake Engineering and Structural Dynamics 29*, 927-944.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California